UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DESPLIEGUE DE IWSN PARA LA DIGITALIZACIÓN EN INDUSTRIA 4.0

David Muñoz Clemente

Tutor: Víctor Miguel Sempere Payá

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2020-21

Valencia, 4 de julio de 2021

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO. ES-CV-000030

Resumen

Las redes LR-WPAN (Low Rate Wireless Personal Area Network) se han convertido en una de las herramientas fundamentales dentro del ecosistema de digitalización del mundo IoT, dadas sus grandes ventajas de flexibilidad, sencillez y bajo coste. Este tipo de redes deben presentar ciertas garantías de robustez para ser utilizadas en escenarios industriales, donde las IWSN (Industrial Wireless Sensor Networks) sirven de mecanismo para nutrir a sistemas como BigData en los entornos de Industria 4.0. Para contribuir en el desarrollo de esta tecnología, se propone simular y analizar una red IWSN basada en IEEE 802.15.4, cuyos mecanismos de acceso permiten garantizar un cierto nivel de garantía en escenarios más agresivos como el industrial. Para ello, se realizarán tareas relacionadas con la planificación en redes sincronizadas y análisis de resultados.

Palabras clave: IIoT, LR-WPAN, IWSN, TSCH

Resum

Les xarxes LR-WPAN (Low Rate Wireless Personal Area Network) s'han convertir en una de les ferramentes fonamentals dins de l'ecosistema de digitalització del món IoT, donades les seues grans ventatges de flexibilitat, sencillesa i baix cost. Este tipus de xarxes han de presentar certes garanties de robusteza per a ser utilitzdes en escenaris industrials, on les IWSN (Industrial Wireless Sensor Networks) serveixen de mecanisme per a nutrir a sistemas com BigData en els entorns d'Industria 4.0. Per a contribuir en el desenvolupament d'aquesta tecnología, es proposa simular i analitzar una xarxa IWSN basada en IEEE 802.15.4, els mecanismes d'accés de la qual permeten garantir un cert nivel de garantia en escenaris més agresius com l'industrial. Per a això, es realitzaran tasques relacionades amb la planificació en xarxes sincronitzades i anàlisi de resultats.

Paraules clau: IIoT, LR-WPAN, IWSN, TSCH

Abstract

LR-WPAN networks (Low-Rate Wireless Personal Area Network) have become in one of the most important tools in the IoT world, thanks to the flexibility, easiness, and low cost. This type of networks must show *guarantees* to be used in industrial scenarios, where the IWSN (Industrial Wireless Sensor Networks) are used in other systems, such as BigData in Industrial 4.0 environments. In order to contribute to the development of this technology, simulation and analysis of a IWSN network based on IEEE 802.15.4 is proposed, whose access mechanisms guarantee a good level of reliability in more aggressive scenarios, such as the industrial. To do so, tasks related with the scheduling process in syncronised networks will be done and analysis of results.

Keywords: IIoT, LR-WPAN, IWSN, TSCH

Índice

Índice de Figuras

Capítulo 1. Introducción, objetivos y metodología

1.1 Introducción

[1]La Industria 4.0, *Industry 4.0* o Cuarta Revolución Industrial, se refiere al proceso de cambio que ha experimentado el modelo de industria, que permite su transformación, digitalización e integración de todos los procesos que conforman la cadena de valor. Más concretamente, la Industria 4.0 afecta directamente al modo en cómo los profesionales están siendo orientados para atender la demanda de los nuevos medios tecnológicos en los próximos años.

[1] El concepto mencionado se fundamenta en la idea de la producción inteligente**,** un nuevo enfoque en los procesos de fabricación: la fabricación computarizada. Adicionalmente, a estas ideas se suman otros procesos anteriores, como el IoT (*Internet of Things*) o el BigData.

En [1] y [7] se definen las cinco ideas principales en las que se fundamenta esta nueva era de la industria, y que definen la sistemática de producción inteligente de los próximos años. Estos principios son:

- **Capacidad de operación en tiempo real:** con tratamiento y adquisición de datos instantáneos y toma de decisiones en tiempo real.
- **Virtualización:** se requiere de una copia virtual de las fábricas inteligentes que permitan el rastreo y monitorización remota de los procesos que se llevan a cabo mediante la instalación de sensores distribuidos.
- **Descentralización:** se debe poder configurar secciones de las infraestructuras, ya que las tomas de decisiones y el mantenimiento de las instalaciones o máquinas no deben afectar todo el proceso en su conjunto.
- **Orientación de servicios**
- **Modularidad:** se debe poder modificar el ritmo de los distintos procesos que se gestionen dependiendo los posibles cambios en los niveles de requerimientos.

Como se ha mencionado, el avance y el progreso de la Industria 4.0 se apoya en los avances tecnológicos de los últimos años, fundamentalmente en las áreas de desarrollo de información e ingeniería. Los campos que han adquirido más relevancia son el *Internet of Things* (IoT) y el BigData.

En [2] definen el IoT como la interconexión de dispositivos y objetos a través de la red (puede ser tanto Internet como una red privada), donde todos estos aparatos son capaces de interactuar entre ellos e intercambiar información. Los dispositivos a los que se hace referencia pueden ser de cualquier ámbito de aplicación, ya sea electrodomésticos, sensores de temperatura, farolas en las calles de las ciudades, dispositivos mecánicos, etc. Como idea global, [2] se pretende que cualquier objeto que se pueda imaginar se conecte a Internet (o como ya hemos dicho a una red privada) sin tener la necesidad de la intervención humana; es decir, una intervención *Machine-to-Machine (M2M).*

El término IIoT (Industrial Internet of Things) es el resultado de la aplicación de IoT en la Industria. Según [3] esta subcategoría del IoT es una de las aplicaciones más importantes, y consiste en la conexión de distintos dispositivos (normalmente sensores) a internet para la toma de decisiones. La diferencia entre IIoT y el IoT es que el primero está principalmente pensado para entornos cerrados.

En [4] definen el BigData como "el conjunto de datos e información cuya velocidad de crecimiento, variabilidad y volumen hacen más compleja su gestión y análisis mediante el uso de tecnologías convencionales". Aunque no está definido el límite a partir del cual se considera BigData, según varios analistas podemos referirnos a datos de volumen entre 30-50 TeraBytes, aunque también empieza ya a hablarse de varios PetaBytes.

[3] Tanto el IoT como el BigData requieren de unas prestaciones muy precisas a nivel de temporización o fiabilidad, debido a las aplicaciones a las que dan uso, como en escenarios de la domótica o aplicaciones industriales, donde la automatización y la programación de la producción han permitido un gran avance en estos campos. Por esta razón, las redes y los dispositivos que formen parte de dichos procesos deben presentar gran robustez y su configuración debe ser rigurosa en cuanto a rendimiento se refiere, asegurando un retardo mínimo, alta fiabilidad y un consumo de energía eficiente.

Las redes WSN (Wireless Sensor Networks) han supuesto gran desafío para su integración en la red, debido a sus características de bajo consumo y bajo coste. Por esa razón, numerosos organismos de estandarización y grupos de investigación han diseñado varios protocolos para la integración de estos dispositivos en las funcionalidades de red como Internet. Estos protocolos mencionados se caracterizan por ser distribuidos, y la funcionalidad principal de ellos es para direccionamiento o enrutamiento del tráfico en la red.

1.2 Objetivos

Los objetivos que se pretenden conseguir con el presente proyecto son:

- Análisis y comprensión de la arquitectura de las redes inalámbricas de sensores (WSN), su funcionamiento y sus aplicaciones más relevantes actualmente.
- Descubrimiento de los protocolos y estándares que rigen las redes WSN y su funcionamiento.
- Comprensión y estudio del protocolo TSCH, las planificaciones y los distintos algoritmos de planificación.
- Prototipado y análisis de prestaciones de redes WSN con diferentes algoritmos de planificación mediante emulación de dispositivos reales en un entorno de simulación.
- Comparativa de algoritmos de planificación TSCH estandarizados.

1.3 Metodología

El desarrollo del proyecto se ha dividido en las siguientes etapas diferenciadas:

1. **Documentación:** mediante la lectura de distintas fuentes bibliográficas y artículos publicados en distintas revistas, se investiga todos los estándares ya existentes sobre la tecnología de las redes LR-WPAN, así como todo lo relacionado con los requisitos y evolución actuales; toda la información del protocolo MAC TSCH y todas las investigaciones realizadas con respecto a los distintos métodos de planificación en este protocolo.
2. **Instalación de software:** instalación de las distintas herramientas de simulación disponibles para el desarrollo y el análisis de las distintas pruebas a realizar.
3. **Desarrollo de pruebas:** lanzamiento de las simulaciones con los distintos parámetros para su valoración.
4. **Análisis de resultados:** evaluación de los resultados obtenidos de las simulaciones para la comparativa de los casos realizados.

La memoria del trabajo se estructura en las siguientes secciones: en los capítulos 2 y 3 se analizará el marco teórico relativo al objeto de estudio del TFG, con las redes WSN y el estándar IEEE 802.15.4. Los capítulos 4 y 5 muestran información relativa al desarrollo de las pruebas y las simulaciones y las distintas configuraciones. Finalmente, en el 6 se detalla el análisis de los resultados obtenidos y en el 7 se exponen las conclusiones y propuestas de trabajo futuro. La relación de bibliografía consultada se presenta en último lugar en la última sección de este documento.

Relación de tareas

Se presenta a continuación una lista de las tareas en las que se ha dividido el desarrollo del proyecto.

- T1: Lectura de artículos de revista, publicaciones de congresos, documentos de Internet y material bibliográfico sobre el auge de las redes WSN, el funcionamiento y el Internet de las Cosas (IoT)
- T2: Lectura e investigación del protocolo MAC TSCH, su funcionamiento y características; algoritmos de planificación y comparativas.
- T3: Instalación del software de simulación en el ordenador, y la configuración de los entornos y todas las dependencias necesarias.
- T4: Desarrollo de simples pruebas para la familiarización con el simulador.
- T5: Desarrollo de script en MATLAB para planificación TSCH sin colisiones.
- T6: Distribución de la red en el simulador.
- T7: Realización de las pruebas y simulaciones con Orchestra, Minimal y la planificación según el algoritmo de AMUS.
- T8: Implementación de funciones en Python para análisis de resultados.
- T9: Obtención de gráficas y comparación de resultados.
- T10: Redacción de la memoria.

Capítulo 2. Redes WSN

Nos referimos a redes de sensores inalámbricos, o WSN (*Wireless Sensor Networks*), a las redes que están constituidas por un conjunto de dispositivos ubicados a lo largo de una superficie y que monitorizan los datos del entorno, como puede ser temperatura, humedad, presión, etc.

El funcionamiento es sencillo: los sensores (nodos) recogen información sobre uno o varios eventos en el núcleo. Los nodos se comunican entre sí para transmitir la información hasta el nodo SINK, siendo este el que transferirá la información a través de la red hasta el destino, generalmente un servidor de datos o aplicaciones.

Existen infinidad de aplicaciones de las WSN, entre las que se pueden destacar:

- **Aplicaciones militares:** despliegue de sensores para detección de movimiento de vehículos en campo de batalla.
- **Aplicaciones médicas:** monitorización de constantes vitales de un paciente u otros parámetros de interés.
- **Aplicaciones industriales:** destacamos en este campo las IWSN (Industrial Wireless Sensor Network), que se caracterizan por su robustez y grandes prestaciones.

  El principio fundamental que siguen este tipo de redes es el ahorro de batería y la optimización del consumo. En general, pero sobre todo en las aplicaciones industriales, los dispositivos son colocados en posiciones generalmente difíciles de alcanzar, por lo que el principal objetivo es el ahorro de la batería y la optimización del consumo de energía (y la potencia).

2.1 Estructura y arquitectura

Como se ha mencionado, la estructura básica de una WSN se compone de un *core* de dispositivos que recogen, monitorizan y transmiten la información del área en el que están instalados; un nodo *sink* que recibe los datos recogidos y transmitidos desde el core y los retransmite a través de una red (generalmente Internet, aunque puede ser otra) para que estos lleguen a su destino, que en la mayoría de los casos se trata de un servidor central de datos. Un diagrama de estructura de una WSN es el que se muestra en la Figura 1:

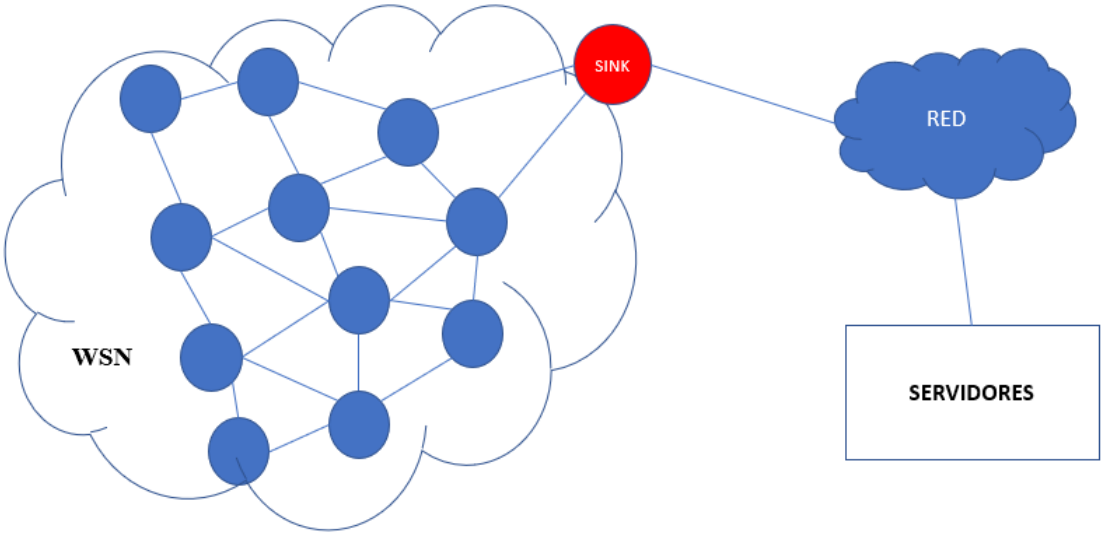

Figura 1: Diagrama de representación de arquitectura de una WSN

En relación con el control y la gestión, diferenciamos el control centralizado, el control descentralizado y el distribuido.

a) **Control centralizado:** un único nodo conoce a modo global el funcionamiento de la red y decide la funcionalidad de los nodos. Es decir, si este debe activarse o no para transmisión o recepción de paquetes.

b) **Control descentralizado:** los nodos de la red se dividen en varios "grupos", y dentro de cada agrupación tenemos un nodo "central" que se comunica con otros nodos centrales de otros grupos.

c) **Control distribuido:** no existe la figura de "nodo central". Por tanto, todos los nodos interactúan entre ellos para la gestión de la red.

En la Figura 2 se muestra un esquema representativo de cada tipo de control descrito.

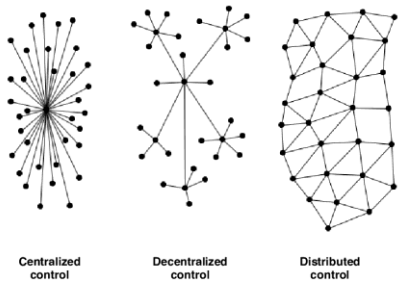

Figura 2: [5] Diagrama de WSN con control centralizado (a), control descentralizado (b) y control distribuido (c)

Bien se ha observado que la forma en la que se dispongan los dispositivos a lo largo de un área es muy importante, ya que puede influir directamente en el rendimiento. En este sentido, podemos diferenciar entre las topologías estructuradas y las topologías no estructuradas.

2.1.1 Topologías Estructuradas

Estas se caracterizan por tener un bajo número de nodos establecidos y, por consiguiente, su gestión es mucho más sencilla. Así mismo, la posición de cada nodo se ha planificado y calculado con anterioridad para optimizar los parámetros de transmisión.

Ejemplos muy comunes dentro de esta clasificación son la disposición en estrella (existe un nodo central y los demás están dispuestos a su alrededor) o en árbol (un nodo principal se encuentra en un nivel "jerárquico" superior y los demás están localizados en otros niveles inferiores). Esquemas ilustrativos de estos ejemplos son los que se encuentran en la Figura 3:

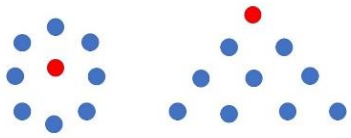

Figura 3: Esquema de representación de topología en estrella (a) y en árbol (b)

2.1.2 Topologías no estructuradas

Estas se caracterizan porque la disposición de los distintos nodos no está prefijada y norma una forma como tal. Es por esta razón que las tareas de gestión y control que deben llevarse a cabo son más costosas en estos casos.

2.2 Protocolos y estándares

Ya se ha mencionado en la introducción del Capítulo 2 que uno de los principales objetivos que se persiguen en estas redes es la optimización del consumo de potencia. En este sentido, existen varios protocolos que regulan el funcionamiento de los distintos ámbitos de las WSN o las redes LR-WPAN, como son el IEEE 802.15.4 [6], Zigbee, 6LoWPAN o ISA100.11a. A continuación, se mencionarán los principales detalles de los tres últimos, y el primero será detallado en el Capítulo 3.

2.2.1 Zigbee

[5] *Zigbee* es el nombre del conjunto de protocolos para la comunicación inalámbrica para dispositivos de bajo consumo, y se basa en el IEEE 802.15.4. El objetivo es el mismo que este último: aplicaciones con bajo requerimiento de tasa de datos y una optimización del consumo energético (intentar alargar el tiempo de vida de las baterías de los dispositivos), pudiendo los sensores monitorizar eventos durante largos periodos de tiempo gracias al estándar [7].

2.2.2 6LoWPAN (IPv6-based Low power Wireless Personal Area Networks)

[5]Este protocolo de nivel de red permite operar con IPv6 sobre el IEEE 802.15.4, y por tanto los nodos pueden comunicarse con otros dispositivos que también operen con IPv6. Tenemos un nivel que adapta los paquetes IPv6 en paquetes del 802.15.4.
El uso más común de 6LoWPAN es en dispositivos embebidos para la automatización del hogar o cuidados de la salud, entre otros.

2.2.3 ISA100.11a

Diseñado para aplicaciones de baja carga de tráfico y monitorización. Además, define el modelo de interconexión para sensores inalámbricos (*Open Systems Interconnection – OSI*). Los principales objetivos que se pretende conseguir son la escalabilidad, bajo consumo de energía y la interacción con otros dispositivos. Incluye además un mecanismo de seguridad para la protección de los datos. A nivel físico, opera también en la banda de 2.4 GHz.

2.2.4 Protocolo RPL

[8] [9] Definido en la RFC5867, RFC5826, RFC5673, RFC5548, el protocolo RPL es un protocolo de enrutamiento para redes inalámbricas de bajo consumo de energía y con susceptibilidad de pérdidas de paquetes. Es un protocolo basado en vectores de distancia y opera sobre el estándar IEEE 802.15.4. Aunque está optimizado para topologías multi-salto y comunicación many-to-one (multipunto-punto), también es viable en redes one-to-one (punto a punto).

Funcionamiento: RPL crea una topología, llamada DAG (Directed Acyclic Graph), similar a un árbol. Los nodos de la red tienen asignados un rango, y este valor aumenta a medida que éstos se alejan del nodo raíz (llamado DODAG). De esta forma, el encaminamiento de los paquetes en cada nodo vendrá definido por el criterio de selección de la ruta con el rango más bajo (el rango de una ruta corresponde a la suma de los rangos de los nodos que la componen).

Paquetes ICMPv6 asociados
En [8] se definen tres tipos de paquetes ICMPv6 asociados al protocolo RPL:

- **DIS (solicitud de información del DODAG):** usado para saber información sobre los nodos raíz (DODAG) cercanos.
- **DIO (objeto de información del DAG):** lleva información de la topología (DAG), y es la respuesta a mensajes DIS. Además, se envía periódicamente para actualizar información de la red.
- **DAO (objeto de actualización del destino):** se envían por los nodos para la actualización de la información de los nodos de rango superior (más cercanos al nodo raíz) a lo largo de la topología.

Capítulo 3. IEEE 802.15.4

[6] El IEEE 802.15.4 es un estándar que define los protocolos del nivel físico y del nivel MAC (*Medium Access Control*- Control de Acceso al Medio) para las redes LR-WPAN (*Low-Rate Wireless Personal Area Network*), redes inalámbricas de baja carga de tráfico, como pueden ser las redes WSN. Aunque este estándar no estaba pensado para dichas redes (WSN), se observó que era bastante adecuado para estas, ya que los principales requerimientos que se mencionan con respecto a las WSN se consiguen en su amplitud en el IEEE 802.15.4.

En [6] y [10] se enumeran varios aspectos fundamentales que conciernen a este estándar:

3.1 Topologías de red

En la versión original de este protocolo, se definieron dos tipos de topologías básicas: topología en **estrella** y topología *peer-to-peer***.**

Para la topología en **estrella**, encontramos a un solo nodo que actúa como PAN Coordinator (*Personal Area Network Coordinator*). En este caso, nos encontramos ante un carácter centralista de las comunicaciones. Es decir, cualquier dispositivo que quiera unirse y transmitir datos en una red de estos tipos, debe enviar sus datos al coordinador y éste enviará el destino correspondiente. Por esta razón, el estándar aconseja que el coordinador debe ser de alto rendimiento de potencia debido a la gran carga de trabajo que debe asumir.

Como consecuencia, y debido a los requerimientos de las WSN que se han mencionado anteriormente, la topología en estrella no es adecuada para su uso en el campo de las redes de sensores. El 802.15.4 recomienda aplicaciones con esta topología como automatización del hogar o juegos y juguetes.

La topología ***peer-to-peer*** presenta mejores prestaciones con respecto a la estrella, a lo que consumo de recursos se refiere. En este caso, encontramos un descentralismo del tráfico, y cada nodo de la red puede comunicarse con cualquier otro, siempre que esté en su rango de alcance.

La Figura 4 muestra un esquema representativo de cada tipo de redes descrito.

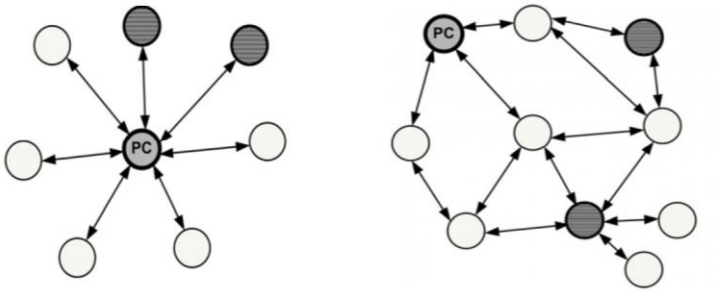

Figura 4: [6] Ejemplo de diagrama de red IEEE 802.15.4 en estrella (a) y *peer-to-peer* (b)

3.2 Capa física

[6]El nivel físico se encarga de la transmisión y recepción de datos en un canal de radio determinado.
En el 802.15.4 se definen tres bandas de frecuencia operacionales, cada una con sus respectivos canales: 2.4 GHz, 915 MHz y 868 MHz. La distribución de los distintos canales se muestra en la Figura 5:

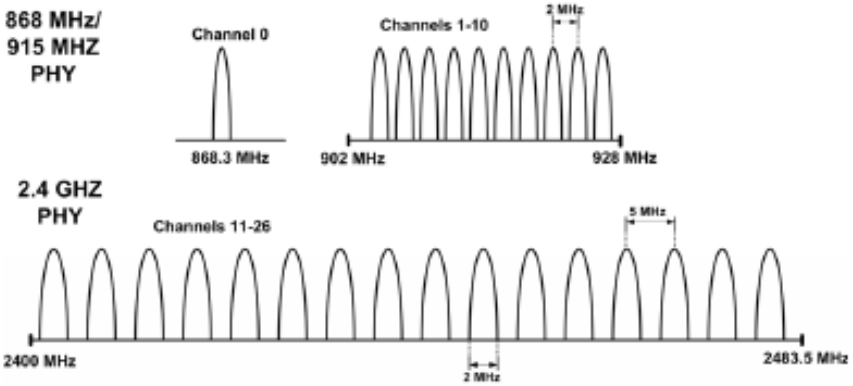

Figura 5: [6] **Distribución de las bandas de frecuencia para el IEEE 802.15.4**

El sub-nivel MAC del 802.15.4 proporciona una interfaz entre la capa física y otras capas (superiores) con otros protocolos para redes LR-WPAN.

A modo general, el estándar IEEE 802.11 (estándar para las redes WIFI) comparte características con el 802.15.4, como el uso del protocolo de acceso al medio CSMA/CA (*Carrier Sense Multiple Access/Contention Avoidance*)

Por último, en [6] se definen dos modos operacionales para la capa MAC, que debe elegir la figura del coordinador de la red. Estos modos son *Beacon-enabled* y *Non Beacon-enabled.*

En el modo *Beacon-enabled, l*os mensajes Beacon son generados periódicamente por la figura del coordinador, y estos sirven para la sincronización de nodos unidos a la red y para la identificación del PAN.

Se menciona la trama beacon, que corresponde a la primera parte de una supertrama, que puede alojar tanto transmisiones de mensajes beacon como transmisión de datos entre nodos.

En cambio, en el modo *non-beacon-enabled,* no tenemos el concepto de supertrama, y los nodos pueden transmitir datos por medio del protocolo CSMA/CA.

3.3 Limitaciones

Todas las prestaciones de ambos modos explicados (Beacon-enabled, Non-Beacon-Enabled) han sido analizadas y estudiadas en profundidad en varios artículos. En [10] se destacan las siguientes limitaciones:

- **Fiabilidad de la comunicación:** tanto en un modo como en otro, debido al funcionamiento del CSMA/CA como protocolo MAC y su ineficiencia. Es evidente que, si el número de nodos de la red es bastante elevado, aumenta el número de colisiones ya que hay varios intentos de transmisión en el canal
- **Sin protección contra las interferencias:** el protocolo MAC del IEEE 802.15.4 usa solo un canal y no hay ningún mecanismo para la protección de interferencias, fenómenos muy comunes en redes inalámbricas.
- **Restricción de energía de los nodos**

Por estas limitaciones mencionadas se considera el IEEE 802.15.4 no adecuado para aplicaciones donde los requerimientos de temporización (obtener el menor retardo posible) y de fiabilidad son muy exigentes. En este sentido, en el año 2008 fue creado el grupo de trabajo 4e, con el principal objetivo de mejorar y corregir las limitaciones descritas. De este equipo de trabajo nació el IEEE 802.15.4e.

3.4 IEEE 802.15.4e

[10] El estándar IEEE 802.15.4e (IEEE 802.15.4e MAC Enhancement Standard) es un estándar aprobado en el año 2012. En este documento se presentan nuevas configuraciones para los distintos nodos de una red WSN, basándose en la anterior versión 802.15.4, y también en otras ideas de otros estándares de aplicaciones industriales (Wireless HART, ISA 100.11.a).

El 802.15.4e presenta nuevos protocolos MAC, cada uno dirigido a diferentes dominios de aplicación. Estos protocolos son: TSCH, DSME, LLDN, AMCA, BLINK. En las siguientes subsecciones se detallarán los tres primeros (debido a que su dominio de aplicación está dirigido a industria y automatización, objeto de estudio del presente proyecto), pero se incidirá más en detalle en TSCH [11].

3.4.1 DSME (Deterministic and Synchronous Multi-channel Extension)

Está diseñado para uso en la industria y aplicaciones comerciales, con exigentes requerimientos a nivel de temporización y fiabilidad. En relación con el acceso al medio, utiliza TDMA, y está especialmente diseñado para redes multi-salto y redes malladas.

3.4.2 LLDN (Low Latency Deterministic Network)

Este protocolo está diseñado para redes *single-hop* y de un solo canal, en especial para automatización de las fábricas, ya que las aplicaciones en ese campo se requieren de una baja latencia.

3.4.3 TSCH (Time-Slotted Channel Hopping)

[11] TSCH (*Time Slotted Channel Hopping*) es un protocolo de la capa MAC (de acceso al medio). El IEEE 802.15.4 se fundamenta en CSMA-CA, por lo tanto, si dos o más nodos intentan realizar una transmisión en el mismo canal en el mismo instante de tiempo, se producirá una colisión y por tanto se deberá producir una retransmisión, lo que conlleva un gasto adicional de energía, lo que no se perseguía con los protocolos que se han explicado.

TSCH se basa en la planificación de la transmisión. En cada nodo se configura cuándo debe transmitir y/o recibir, y por tanto encender la radio. Dicha planificación se realiza mediante una matriz o tabla, en la que cada fila se corresponde a un Channel Offset (un canal distinto para evitar interferencias) y cada columna a un timeslot (una ranura temporal). Un slotframe está compuesto de una serie de timeslots, y una macrotrama está compuesta por una serie de slotframes. Un modelo de tabla o matriz de planificación es el que se muestra en la Figura 6 en la que hay 4 Channel-Offset y 7 timeslots:

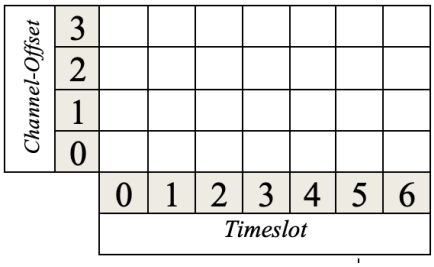

Figura 6: Ejemplo de modelo de planificación TSCH

Cada celda de la tabla se corresponde con un enlace de comunicación entre dos nodos adyacentes. Tan solo un nodo debe transmitir en una única celda (en el mismo timeslot), ya que de no ser así se produciría una colisión. Podemos diferenciar dos tipos de celdas:

- **Celdas compartidas:** para la transmisión de paquetes de broadcast (como pueden ser paquetes Beacon o paquetes de RPL). Un nodo transmite a todos los nodos adyacentes y estos están configurados para la recepción de los paquetes.
- **Celdas dedicadas:** para transmisiones unicast. Un solo nodo transmite a otro nodo.

El ***Channel-Offset*** es un pseudo-canal, definido para indicar en qué canal físico real debe producirse la comunicación. Para la obtención de dicho canal (real), se aplica la ecuación (1):

$$Canal\ Físico = F\{(ASN + Channel_{Offset})\ mod\ C\} \qquad (1)$$

Donde:

- F{…} es una función biyectiva
- ASN (*Absolute Slot Number*) es el número de timeslot absoluto
- C es el número de canales físicos disponibles.

Aunque este nuevo protocolo proporciona nuevas ventajas a nivel de fiabilidad, el estándar no define las formas o reglas de cómo crear y mantener una buena planificación de la transmisión, que es en donde reside la importancia a la hora de usar TSCH como protocolo MAC. Existen varios estudios y varias propuestas de métodos de planificación, que se pueden clasificar en centralizados, autónomos o distribuidos.

En los métodos centralizados, una entidad o nodo central elabora se encarga de elaborar la planificación a través de la información que ha recibido de los nodos de la red. Típicamente esta información proviene de mensajes del protocolo de control RPL (2.2.4) que informan sobre la distribución y las adyacencias de cada nodo.

[12] Los algoritmos de planificación TSCH autónomos se caracterizan por la realización de la planificación con la información relativa sobre la topología y el encaminamiento (aunque no se trata exclusivamente de este tipo de información [13]) que han intercambiado con sus "vecinos" y con los "hijos", según los mensajes de control RPL que han sido transmitidos y/o recibidos. Por lo tanto, no necesita de señalización adicional para la conformación de las distintas planificaciones y solo es necesario la dirección MAC e información de topología actualizada [12]. En [13] se definen 6 aspectos en los que se fundamentan las propuestas de planificación autónomas: interacción con el encaminamiento (1), alojamiento de celdas (2), alojamiento de canales (3), caso de aplicación (4), realojamiento de celdas (5) y adaptación al tráfico (6).

El algoritmo autónomo con más renombre es Orchestra (0) [14] [15], aunque en [13] se detallan y explican otras propuestas, entre las que destaca ALICE (Autonomous Link-based Cell Scheduling for TSCH) [12]. En [12] se presenta el funcionamiento y diseño de este nuevo algoritmo y se mencionan las diferencias y ventajas que presenta este sobre Orchestra, entre las que destaca el mejor rendimiento en cuanto a fiabilidad, retardo y congestión debido a la naturaleza "link-based" frente al carácter "node-based" de Orchestra, demostrando la pérdida de recursos en planificaciones realizadas por este (Orchestra).

Por último, los algoritmos distribuidos consisten en la negociación de las planificaciones de cada nodo con los vecinos para el cómputo de estas. En términos de fiabilidad y retardos las prestaciones son similares a los métodos centralizados [16], pero en este caso tenemos un coste añadido por la señalización y mensajes de control.

En [16] consideran 3 posibles alternativas para la asignación de una celda de un nodo con uno adyacente: (1) compartir una misma celda entre todos los nodos, lo que reduce la complejidad de la señalización de las planificaciones pero tiene efectos negativos cuando hablamos de redes más extensas debido a las interferencias posibles; (2) elección de un timeslot según la información de mensajes, lo que provoca un aumento del consumo de energía; (3) elección de un timeslot arbitrario, opción para la que es necesario el uso de funciones resumen (hash) para calcular las coordenadas de la celda que asignarán dos nodos adyacentes. La elección de la estrategia de asignación vendrá determinada por los requerimientos de la aplicación de uso: [16] [17]para una aplicación con requerimientos de fiabilidad y alta carga de tráfico se elegirá la opción de señalización adicional, mientras que, si hay más requisitos en cuanto a temporización, se recomienda el uso de las funciones resumen. No obstante, en los últimos años se han propuesto algoritmos híbridos que usan ambas estrategias (mensajes de control y hash), como DIS_TSCH [17].

En las próximas secciones se describirán algunos de los métodos de planificación: TSCH-Minimal, Orchestra y AMUS.

3.5.3.1 TSCH-Minimal

La planificación TSCH-Minimal es simple y debe ser compatible con todos los nodos disponibles. Se trata de un algoritmo distribuido y que destaca por su sencillez.

La planificación Minimal consiste en un solo slotframe, con una sola celda compartida por todos los nodos (generalmente esta celda se ubica en el timeslot 0 y el channel-offset 0). Un ejemplo de planificación minimal se muestra en la Figura 7, para un slotframe de 7 timeslots y 4 channel-offsets, en el que la celda compartida se encuentra en el timeslot 0 y el channel-offset 0.

0							
1							
2							
3							
	0	1	2	3	4	5	6

Figura 7: Ejemplo de planificación TSCH-Minimal

3.5.3.2 Orchestra

[15] El método Orchestra es otro método de planificación en el que cada nodo crea y mantiene su propia planificación y la va actualizando a medida que se suceden cambios en la red. De hecho, utilizan los mensajes del protocolo RPL (ver 2.2.4) recibida por parte de los vecinos y de los padres.

Una planificación Orchestra se compone de varios slotframes, cada uno de ellos dirigido para un tipo de tráfico específico: mensajes Beacon (broadcast), mensajes RPL (DIS, DAO, DIO) (ver 2.2.4) o mensajes unicast (mensajes de aplicación). Puede suceder que en la planificación se solapen dos ranuras de slotframes distintos (tipo de tráfico distinto) y haya conflicto de transmisión. En ese caso, transmitirá la celda del tráfico con mayor prioridad.

Un ejemplo de lo explicado en el párrafo anterior se observa en la Figura 8:

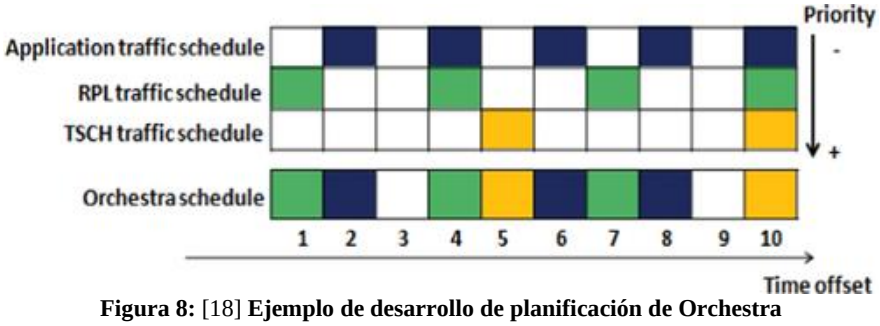

Figura 8: [18] **Ejemplo de desarrollo de planificación de Orchestra**

En los 3 primeros slots no hay conflictos. Sin embargo, vemos que en el timeslot 4 hay dos celdas en conflicto correspondientes a la planificación del tráfico de aplicación y la planificación de RPL. Al tener este último mayor prioridad frente al primero, la celda asignada en planificación de *Orchestra* es la de RPL. Caso similar se encuentra en el timeslot 10, en el que también hay tráfico de TSCH, siendo este el de mayor prioridad y por tanto la celda es asignada en la planificación de *Orchestra.*

Para planificaciones con Orchestra, en [15] se definen varios tipos de slots:

- **Common Shared Slot:** una celda para uso en transmisión y/o recepción.
- **Receiver-based Shared Slot:** cada nodo tiene asignada una celda en recepción y una celda en transmisión por cada nodo vecino.
- **Sender-based Slot:** encontramos un slot en transmisión para cada nodo y uno en recepción para cada vecino.

En este sentido, los tipos de slotframes que pueden conformar una planificación Orchestra se detallan a continuación:

- **EB Slotframe:** se trata de un slotframe formado por slots Sender-based dedicadas, y el tipo de tráfico que se transmite es EB (Enhanced Beacon).
- **Broadcast Slotframe:** se conforma de slots de tipo Common-Shared, principalmente para tráfico RPL broadcast.
- **Receiver-based Unicast Slotframe:** para tráfico unicast, formada por slots receiver-based.
- **Sender-based Unicast Slotframe:** para tráfico unicast, formada por slots sender-based.

En resumen, una planificación Orchestra se puede realizar en función de la configuración que se decida: sender-based o receiver-based:

a) **Configuración Sender-based:** conforma la planificación con un EB Slotframe, un Broadcast Slotframe y un Sender-based Unicast Slotframe.
b) **Configuración Receiver-based:** formada por un EB Slotframe, un Broadcast Slotframe y un Sender-based Unicast Slotframe.

Todos los parámetros que se han explicado y detallado se resumen en la Tabla 1, donde se detallan una serie de reglas de diseño de Orchestra que indican la configuración de los timeslots y la forma en que se calculan.

Tabla 1: Reglas de diseño y configuración de Orchestra

REGLA	SLOTFRAME	MODO	TIMESLOT_TX	TIMESLOT_RX
Default-Common	-Común con enlace compartido. -Unicast & Broadcast	1) Si hay EB → LINK_NORMAL 2) Si no hay EB → LINK_ADVERTISING		
EB-per-time-source	-Dedicado a transmisión de EB		Hash (MAC) % ORCHESTRA_EBSF_PERIOD	Hash (timesource.MAC) % ORCHESTRA_EBSF_PERIOD
Unicast-link-based	-Dedicado a transmission unicast. -Solo a almacenamiento RPL	Link-based	Hash (MAC, nbr.MAC) % SB_UNICAST_PERIOD	Hash (nbr.MAC,MAC) % SB_UNICAST_PERIOD
Unicast-per-neighbor-rpl-ns	-Dedicado a transmission unicast. -Se añade un link por timeslot	Receiver-based	Hash (nbr.MAC) % SB_UNICAST_PERIOD	Hash (MAC) % SB_UNICAST_PERIOD
Unicast-per-neighbor-rpl-storing	-Transmisión unicast -SOLO almacenamiento RPL. -Se añade link *si el slot es necesario*	Receiver-based	Hash (nbr.MAC) % ORCHESTRA_UNICAST_PERIOD	Hash (MAC) % ORCHESTRA_UNICAST_PERIOD
		Sender-based	Hash (MAC) % ORCHESTRA_UNICAST_PERIOD	Hash (nbr.MAC) % ORCHESTRA_UNICAST_PERIOD

3.5.3.3 Algoritmo AMUS

En [19] se define el algoritmo AMUS (*Adaptive MUlti-hop Scheduling*). Es un algoritmo centralizado para planificación TSCH que aporta fiabilidad en cuanto a baja latencia para aplicaciones con requerimientos de temporización bajos, como son la mayoría de las aplicaciones y procesos industriales.

Este método consiste en la reserva de las distintas celdas (recursos) de la planificación a lo largo de las distintas rutas que siguen los paquetes en una red. En este sentido, el algoritmo propone conseguir ese objetivo mediante un proceso que consta de tres fases:

1. **Recolección de información de la red**
   Mediante protocolos de enrutamiento y control, como el RPL, se recoge la información de estadísticas de la red para saber parámetros como el número de paquetes que genera un nodo en un periodo, topologías etc.

2. **Asignación de celdas**
   Para una correcta planificación sin colisiones y con garantías de temporización, es necesario conocer previamente las secuencias de envíos que se van a llevar a cabo, ya que, en la mayoría de los casos, los nodos más cercanos al SINK deben encaminar paquetes de nodos que están más alejados de este. Por tanto, la celda que emula la comunicación del paquete más cercano al SINK a este debe situarse en un timeslot en el que nos aseguremos que ha llegado el paquete a transmitir.

   El proceso a seguir es el siguiente:

   a) **Detalle de las rutas que seguirán los paquetes a lo largo de la red hasta el sink.**
   b) **Construcción de la matriz SS.**

      La matriz SS es una matriz 2xn (n=número de nodos que transmiten paquetes), en la que la primera fila indica el ID del nodo transmisor y la segunda fila la tasa de tráfico de cada nodo. Cada columna de la matriz SS representa un flujo de tráfico.
   c) **Construcción de la matriz MSS**

      La matriz MSS es una matriz de tres filas. Cada columna de dicha matriz representa un enlace entre dos nodos adyacentes, siendo la primera fila el nodo origen, la segunda el nodo destino y la tercera la tasa de tráfico

3. Planificación

Una vez obtenida la matriz MSS, se comienza a asignar en orden de columnas las celdas, desde el primer timeslot libre y el primer Channel Offset. Se asignarán tantas celdas como indique la tasa (tercera fila de MSS).

Las celdas deben asignarse siguiendo tres reglas para evitar colisiones o interferencias en el medio y por tanto se produzcan pérdidas y retardos mayores. [19] Estas reglas son:

a) No puede haber múltiples transmisiones del mismo enlace en el mismo timeslot. Se deben producir secuencialmente (timeslots consecutivos).

b) Las transmisiones de otros enlaces que puedan producir colisiones o interferencias deben realizarse en otro canal de este timeslot (si ya ha sido asignado), o en su caso, en otro timeslot.

c) Los enlaces que no tengan ningún conflicto o ninguna interferencia pueden asignarse de forma paralela: se puede usar el mismo timeslot del mismo canal.

A modo de resumen se proporciona la Tabla 2 en la que se comparan los aspectos más relevantes de los tres métodos de planificación detallados en esta sección:

Tabla 2: Comparación entre los tres métodos de planificación presentados

	AMUS	MINIMAL	ORCHESTRA
Formación/Construcción de la topología	Protocolo RPL	Protocolo RPL	Protocolo RPL
Distribución de planificación	**Centralizado** (una entidad PCE[1] central asigna recursos)	Distribuido	Autónomo (cada nodo genera su propia planificación)
Reserva de recursos	**Específica** (en función de las necesidades de los nodos)	Una única celda	Flexible
Respuesta a cambios/Adaptativo	No. Planificación previa	No. Una única celda planificada	Sí (mensajes RPL)
Evita colisiones	Sí	No. Una celda compartida en el mismo timeslot en el mismo canal.	Sí
Distinción tráfico	No	No.	Sí. 3 slotframes según el tráfico: -EB Slotframe (Beacons) -Broadcast Slotframe (Mensajes RPL) -Unicast-Slotframes (aplicación)

[1] PCE: Path Computation Element

Capítulo 4. Entorno de simulación

Para valorar y comparar los tres algoritmos de planificación presentados en las secciones previas: Minimal, Orchestra y AMUS, se propone la simulación de una red con distintos parámetros de tráfico y así evaluar las distintas prestaciones que ofrecen cada uno de ellos y las ventajas que pueden suponer dependiendo el caso de uso de las aplicaciones. A continuación, se presentará el entorno de simulación Cooja utilizado y las utilidades de este. Posteriormente, se aportarán datos técnicos del modelo de dispositivos que se han emulado y finalmente se darán explicaciones más detalladas de los parámetros de simulación de interés.

4.1 Simulador Cooja

Contiki-NG [20] es un sistema operativo para el estudio y análisis de dispositivos IoT (Internet of Things). Es de código abierto y compatible con sistemas Linux, aunque se puede adaptar a otros sistemas operativos mediante herramientas como Docker.

Contiki-NG dispone de Cooja, un entorno de simulación escrito en lenguaje Java y que es capaz de simular y emular redes WSN. A parte de la gran ventaja que supone disponer de un simulador de estas características, cabe destacar la funcionalidad de emulación de dispositivos reales (dispositivos comerciales utilizados en aplicaciones reales), lo que permite obtener unos resultados más realistas y la posibilidad de que estos sean más fiables a la hora del diseño y prototipado de las redes.

Seguidamente, se detallarán por un lado el procedimiento seguido para la instalación del software para la realización de este proyecto; finalmente, se hará una breve descripción de las partes principales y las funcionalidades de Cooja.

4.1.1 Instalación de Cooja en Linux

Si no se dispone de Linux como sistema nativo, se sugiere la instalación de una máquina virtual con mínimo de 4GB de RAM y 30 GB de almacenamiento (disco duro virtual), mediante asistentes como Oracle VM o VM Ware.
En primer lugar, se debe clonar el repositorio de Contiki en la máquina, mediante el comando:

```
$ git clone https://github.com/contiki-ng/contiki-ng.git
```

Una vez clonado el repositorio, será necesario la actualización del mismo:

```
$ cd contiki-ng
$ git submodule –update –recursive
```

Posteriormente, se descarga la imagen de Docker correspondiente al contenedor de Contiki y se arranca. El comando que realiza esa funcionalidad es:

```
$ sudo docker run –privileged –sysctl
net.ipv6.conf.all.disable_ipv6=0 -mount type=bind,
source=/home/user/Escritorio,
destination=/home/user/Contiki-ng -e DISPLAY=$DISPLAY -v
/tmp/.X11-unix -v /dev/bus/usb:/dev/bus/usb -ti
contiker/Contiki-ng
```

Finalmente, con el contenedor Docker ya arrancado, tan solo será necesario acceder al directorio donde se encuentra el código fuente de Cooja y proceder a su compilación y ejecución mediante ant:

```
$ cd contiki-ng/tools/cooja
$ ant run
```

La pantalla de inicio de Cooja es la que se muestra en la Figura 9:

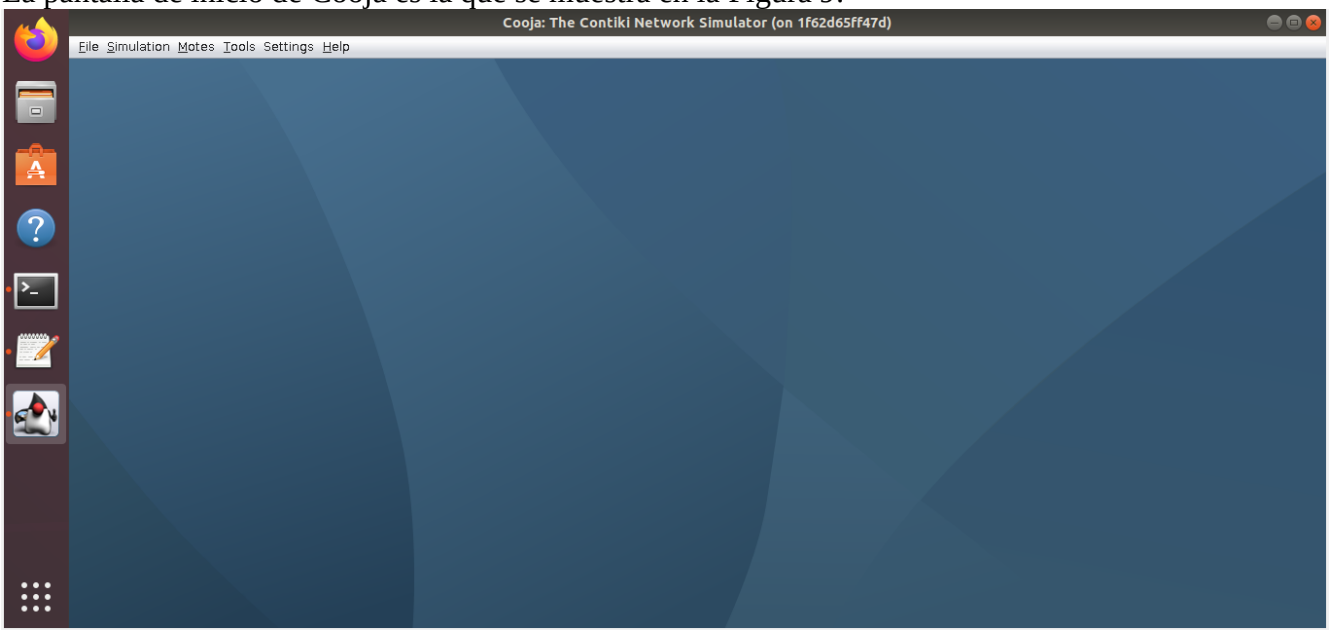

Figura 9: Pantalla de inicio del simulador Cooja

4.1.2 Funcionamiento de Cooja

4.1.2.1 Interfaz de simulación

El funcionamiento de las simulaciones en Cooja es simple y se rige por una serie de ventanas que se muestran por defecto al iniciar una simulación, aunque está a elección del usuario añadir otras ventanas disponibles o eliminar algunas que ya aparecen. En este sentido, se va a detallar a continuación el funcionamiento y la utilidad de las más importantes:

1. **Network graph:** en esta ventana se puede observar un diagrama de la topología de la red que se pretende simular, donde aparecen los IDs de cada nodo añadido. Adicionalmente se puede elegir la opción de visualizar el tráfico (aparecen como flechas azules entre los nodos extremos de los enlaces de comunicación), o el rango máximo de alcance de los dispositivos (alcance de radio).
2. **Radio Messages:** se muestra el contenido de todos los mensajes que se transmiten por toda la red. Existe la opción de exportar los datos para que sean analizados con un analizador de protocolos (en este caso Wireshark, el más conocido).
3. **Timeline:** se muestran los distintos eventos que se suceden a lo largo de la simulación. Los eventos se muestran para cada nodo, y entre estos puede haber: estados de la radio (aparece una línea gris cuando esta está activada), o eventos relativos a las transmisiones (un pequeño trazo azul indica que un paquete está siendo transmitido; uno verde cuando la recepción de un paquete ha sido satisfactoria, o un trazo rojo si se ha producido colisión).
4. **Mote Output:** en esta ventana se muestran mensajes relativos a los eventos que se suceden durante la simulación. Estos mensajes pueden ser generados por el propio código de los nodos que ha sido compilado, o por propios procesos internos de Cooja que provocan mensajes de advertencia o error.
5. **Simulation Control:** en esta ventana es posible el control de la simulación mediante los botones Start (para iniciarla), Pause o Reload (para recargar el código y empezar de nuevo la simulación).

En la Figura 10 se muestra una captura de pantalla con las distintas ventanas explicadas numeradas.

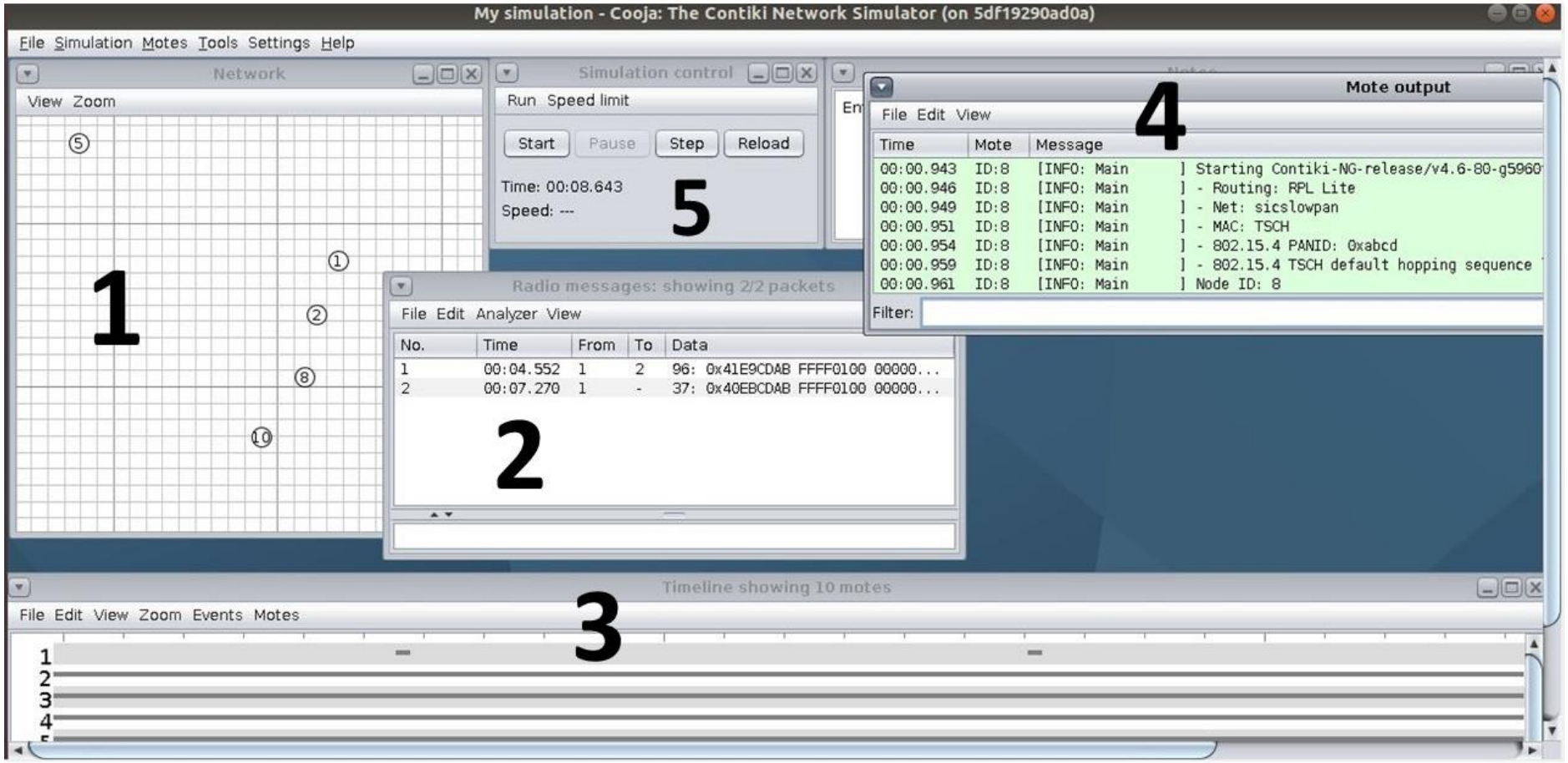

Figura 10: Interfaz de simulación de Cooja con las principales ventanas numeradas

4.1.2.2 Programación de aplicaciones en Contiki

Las aplicaciones que se compilan en Contiki deben seguir una estructura en los archivos de los nodos que se quieren simular. Una de las principales ventajas es que soporta la programación multihilo, por lo que es posible ejecutar varios procesos en paralelo. Las aplicaciones deben ser escritas en lenguaje C, y la estructura que deben seguir es la que se detalla:

```
(Archivos de cabecera)
#include …
#include …

//Define procesos
PROCESS(proceso1, "proceso1");
PROCESS(proceso2, "proceso2");
…
PROCESS(procesoN, "procesoN");

//Desarrolla hilos
PROCESS_THREAD(proceso1, ev, data)
{
   /* Declaración de variables */
   …
   PROCESS_BEGIN();
   …
   …
   PROCESS_END();
}

PROCESS_THREAD(procesoN, ev, data)
{
   /* Declaración de variables */
   …
   PROCESS_BEGIN();
   …
   …
   PROCESS_END();
}
```

4.1.2.3 Archivos adicionales

A parte del archivo del firmware escrito en C (0), se pueden utilizar otros archivos para la configuración de las aplicaciones, como son el archivo `project-conf.h` y el Makefile.

El archivo `Project-conf.h` permite configurar las aplicaciones y simulaciones de Contiki de forma particular para cada caso, y variando los distintos parámetros de los módulos adicionales que se incluyan en la simulación o aplicación.

El archivo `Makefile` permite incluir los diferentes módulos y sub-módulos adicionales que se encuentren en el directorio raíz `os`, en las aplicaciones o simulaciones.

4.1.2.4 Creación de un nodo

Para añadir un nodo a la simulación, una vez que tenemos el código de la aplicación compilado, hay que ir primero al menú: Motes → Create Mote → y seleccionamos el modelo de dispositivo que queremos emular.

La ventana que aparece es como se muestra en la Figura 11:

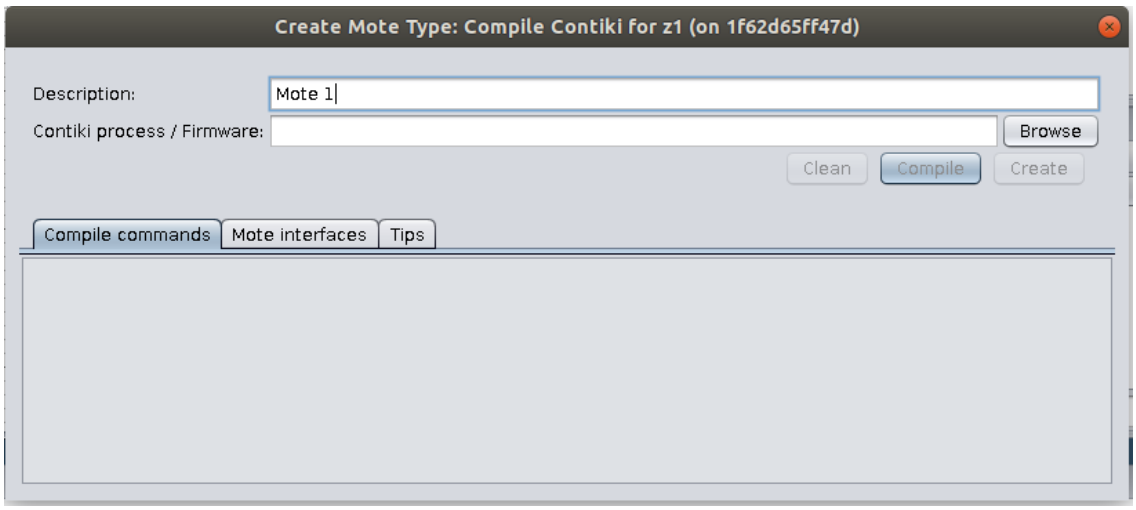

Figura 11: Ventana de creación e inserción de un nodo en la red de la simulación

4.2 Otras herramientas

4.2.1 Simple-Energest

[21] Simple-Energest es un módulo incluido en Contiki que permite el cálculo del consumo de energía de los dispositivos emulados en simulaciones de Cooja. Se fundamenta en calcular el tiempo en el que han estado los dispositivos emulados en los diferentes estados disponibles (CPU activa, transmisión, recepción, etc.). Junto con la información obtenida mediante el módulo, y la información relativa a los consumos de los distintos modos de operación de los dispositivos que se quieran simular (se puede obtener esa información consultando los informes del fabricante) se puede fácilmente calcular el consumo energético de los dispositivos.

Para activar la extensión, hay que añadir la variable en el archivo `Project-conf.h` e importar el módulo en el archivo `Makefile`, tal como se indica en la Figura 12

```
#define ENERGEST_CONF_ON 1

MODULES += os/services/simple-energest
```

Figura 12: Código referente a la configuración de Energest

La configuración por defecto inicia la aplicación al arrancar una simulación, y el periodo de reporte es de 60 segundos (cada 60 segundos se realiza una medición de los datos temporales), aunque se pueden configurar fácilmente en los archivos fuente.

Los datos de salida de la aplicación se muestran en la ventana Mote Output (0), o en su lugar también se puede observar en la interfaz serie de los nodos.

Un ejemplo de mensaje de Energest es el que se muestra en la Figura 13 :

```
[INFO: Energest  ] --- Period summary #2 (60 seconds)
[INFO: Energest  ] Total time  :    1966080
[INFO: Energest  ] CPU         :      10374/   1966080 (5 permil)
[INFO: Energest  ] LPM         :    1955706/   1966080 (994 permil)
[INFO: Energest  ] Deep LPM    :          0/   1966080 (0 permil)
[INFO: Energest  ] Radio Tx    :        106/   1966080 (0 permil)
[INFO: Energest  ] Radio Rx    :     104802/   1966080 (53 permil)
[INFO: Energest  ] Radio total :     104908/   1966080 (53 permil)
```

Figura 13: [21] Ejemplo de mensaje de Energest en un nodo Z1 emulado

Los campos que se destacan son:

- **Número de mensaje**: indica el número de mensaje Energest que se ha detallado.
- **Total time:** indica el número total de ticks que hay en el periodo que se ha reportado.
- El resto de líneas indican el total de ticks que ha permanecido el nodo en el estado que se detalla. En el caso de la Figura 13, el nodo que se ha emulado ha permanecido un total de 10374 ticks en el estado de CPU activa.

Capítulo 5. Desarrollo y pruebas

La red que se propone para la simulación y el análisis de las prestaciones de los métodos de planificación presentados en la sección 3.4.3, es la que se muestra en la Figura 14**¡Error! No se encuentra el origen de la referencia.**, una red de 10 nodos en la que el nodo con ID=1 actúa como nodo SINK.

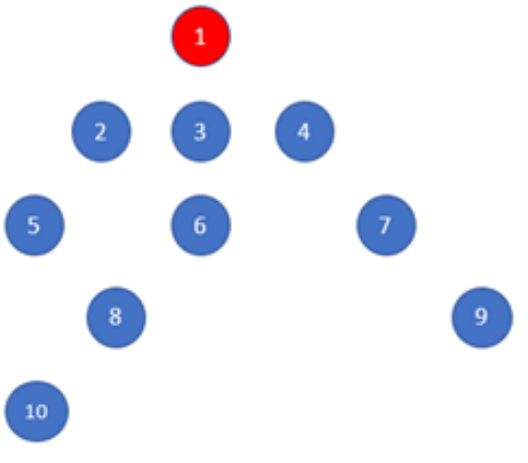

Figura 14: Diagrama de la red analizada

En la Tabla 3 se especifican los parámetros principales de las simulaciones.

Tabla 3: Parámetros de simulación

Tamaño Slotframe	7 Timeslots
Canales	4
Número de paquetes	500
Tiempo de envío de paquetes	70, 140 y 500 ms

Las pruebas se van a realizar sobre dos tipos de nodos (*motes*, en el menú de Cooja), que van a diferenciar el análisis a realizar. En primer lugar, se van a realizar las pruebas con Cooja Motes, son nodos creados específicamente para Cooja y únicamente son virtuales, y no emulan ningún dispositivo real. Por esa razón esta parte de las pruebas es útil para el estudio analítico y teórico de la temporización. Las planificaciones van a ser realizadas por el método Orchestra, Minimal y la planificación como resultado del algoritmo descrito en AMUS. Los mensajes se transmiten sobre UDP, usando las funciones disponibles en Contiki.

Por otro lado, como segunda sección de la parte de pruebas se va a analizar con Z1 Motes. Se trata de un dispositivo compatible con los protocolos IEEE 802.15.4 y ZigBee.

Su núcleo (core) se compone del microcontrolador MSP430 y del transceptor de radio CC2420, del fabricante Texas Instruments. Este hecho permite también que estos dispositivos sean compatibles con sensores de las mismas características y por tanto no haya conflictos en cuanto a código de aplicaciones y firmware se refiere.

Según el informe del fabricante [22], las aproximaciones del consumo de corriente para el módulo de radio (CC2420) y para el microcontrolador (MSP430f2617) son las que se muestran en la Tabla 4, aunque estos datos serán de interés en secciones posteriores para la evaluación de las prestaciones.

Tabla 4: Datos de consumo de los módulos de los nodos Z1

MÓDULO	Rango de operación	Consumo de corriente	Modos de operación
MSP430f2617	1.8V-3.6V	0.1µA	OFF
		0.5µA	ON
		0.5mA	Activo @1MHz
		< 10mA	Activo @16MHz
CC2420	2.1V-3.6V	<1µA	OFF
		20µA	Power Down
		426µA	IDLE
		18.8mA	RX
		17.4mA	TX @0dBm

5.1 Configuración general de los nodos

Teniendo en cuenta lo explicado con respecto a las aplicaciones en Contiki (ver 0), el código del firmware se ha dividido en dos hilos:

1) Proceso principal: en el que se inicia el protocolo RPL y se invoca el segundo proceso para el nodo transmisor (que en el caso de las simulaciones es el nodo 10).
2) Proceso de desarrollo: en el que se realizan todas las interacciones para el proceso de envío de los 500 paquetes planificados.

El código completo utilizado para los nodos se muestra en el ANEXO A.

A pesar de las grandes ventajas y aplicaciones que supone el auge de las redes WSN que se han ido describiendo, un gran inconveniente que se presenta es la capacidad de memoria limitada de los nodos. Por esta razón, hay que ser muy rigurosos con el estilo del código y el uso de la memoria a la hora de programar las aplicaciones. En este sentido, para nuestras simulaciones con los nodos Z1, hemos tenido que seguir algunos consejos y directrices explicados en [20] con respecto a la reducción del uso de la memoria ROM y RAM para que los códigos puedan ser compilados.

En relación con la configuración de TSCH, cabe decir que por defecto Contiki tiene CSMA como protocolo MAC. Por esa razón, hay que cambiarlo mediante el Makefile, añadiendo la siguiente línea:

```
MAKE_MAC=MAKE_MAC_TSCH
```

Adicionalmente, como uno de los parámetros que se pretende analizar es el consumo de energía, ya que como se ha mencionado en secciones anteriores, la optimización del consumo energético

en redes WSN es uno de los principios fundamentales, usaremos el módulo Simple-Energest (ver 4.2.1). Y como ya se ha explicado, para incluirlo en el proyecto hay que añadir una línea adicional al archivo Makefile.

Cabe señalar que el código completo de todos los archivos que se están mencionando, cuyo contenido se hace alguna referencia en esta sección, puede consultarse por completo en el ANEXO A, donde se muestra el programa del firmware (node.c), el fichero Project-conf.h y el fichero Makefile.

A continuación, se detallarán las configuraciones particulares para los tres algoritmos de planificación que se pretende analizar.

5.1.1 Configuración de Orchestra

Orchestra está incluido como módulo en Contiki. Por tanto, para incluirlo en nuestro proyecto hay que añadir la línea correspondiente al fichero Makefile:

```
MODULES += os/services/orchestra
```

Además, será necesario incluir la variable BUILD_WITH_ORCHESTRA definida como 1 en el fichero Project-conf.h para indicar que queremos activar la planificación Orchestra. Como se ha explicado en 3.5.3.2 Orchestra, Orchestra se compone de varios slotframes dependiendo del tipo de tráfico y de diferentes reglas de diseño. Para nuestra simulación, dejaremos la configuración por defecto en relación con los slotframes de tráfico RPL y Enhanced Beacon (longitudes de 31 y 397 timeslots, respectivamente). El único tamaño que se modificará es el unicast (7 timeslots), mediante la línea que se incluye en el fichero de configuración, tal y como se indica en la Figura 15:

```
#define BUILD_WITH_ORCHESTRA 1
#define ORCHESTRA_CONF_UNICAST_PERIOD 7
```

Figura 15: Configuración de Orchestra en Contiki

5.1.2 Configuración de 6TiSCH-Minimal

Para la simulación con planificación Minimal tan solo es necesaria la inclusión de TSCH y la declaración de la variable TSCH_SCHEDULE_CONF_WITH_6TISCH_MINIMAL con valor a 1 en el fichero Project-conf.

5.1.3 Configuración de AMUS

Cabe señalar en este apartado que la planificación con AMUS no se ha hecho tal cual se detalla en [19] con todos los procesos explícitos. Para nuestro caso se ha omitido el elemento del PCE y la recolección de toda la información previa de enrutamiento y su transmisión a este. De hecho, para nuestro caso de simulación solo usaremos el algoritmo de planificación descrito en [19] para la planificación, asumiendo que toda la información ya es conocida, incluidas las rutas que seguirán los paquetes.

Como herramienta adicional, se ha desarrollado un script en MATLAB que recibe la matriz MSS, el número de nodos, canales y timeslots del slotframe para el alojamiento de las celdas en la planificación. Dicho script realiza el pseudoalgoritmo descrito en [19] y muestra a la salida la

planificación, que se garantiza sin colisiones ni interferencias. El script de MATLAB completo se puede encontrar adjunto en el ANEXO B.

De esta forma, la matriz SS asociada a nuestra red es:

$$SS = \begin{bmatrix} 10 \\ 1 \end{bmatrix} \qquad (2)$$

La ruta que seguirán los paquetes es la de menor número de saltos y la más cercana, que es: 10→8→6→3→1. Por tanto, la matriz MSS asociada es:

$$MSS = \begin{bmatrix} 10 & 8 & 6 & 3 \\ 8 & 6 & 3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (3)$$

Tras ejecutar el programa de MATLAB con los parámetros introducidos, la planificación resultante es la que se muestra en la Tabla 5:

Tabla 5: Planificación TSCH obtenida según el algoritmo

Channel Offset	0	Comp	10→8	8→6	6→3	3→1		
	1	Comp						
	2	Comp.						
	3	Comp.						
		0	1	2	3	4	5	6
		Timeslot						

Una vez obtenida la planificación, se configura en los nodos utilizando la API de TSCH de Contiki. En nuestro caso, se ha creado una función dentro del fichero node.c que configura toda la planificación y tan solo será necesario invocarla. Además, hay que declarar la variable 6TISCH_MINIMAL a 0 en el Project-conf.h para que no se realice planificación con Minimal. El código de la función es el que se muestra en la página siguiente:

```
void
tsch_schedule_create_example(void)
{
    struct tsch_slotframe* sf_custom0;
    tsch_schedule_remove_all_slotframes();
    sf_custom0=tsch_schedule_add_slofframe(0,7)

    linkaddr_t address_1; //Direcciones MAC de los nodos
    linkaddr_t address_3;
    linkaddr_t address_6;
    linkaddr_t address_8;
    linkaddr_t address_10;


    //Repetir este proceso para las 5 direcciones

    address_N.u8[0]= 0 & 0xFF;
    address_N.u8[1]= N & 0xFF;
    address_N.u8[2]= 0 & 0xFF;
    address_N.u8[3]= N & 0xFF;
    address_N.u8[4]= 0 & 0xFF;
    address_N.u8[5]= N & 0xFF;
    address_N.u8[6]= 0 & 0xFF;
    address_N.u8[7]= N & 0xFF;

    tsch_schedule_add_link(sf_custom0, LINK_OPTION_TX | LINK_OPTION_RX |
LINK_OPTION_SHARED, LINK_TYPE_ADVERTISING, &tsch_broadcast_address,0,0,0);


    //Asigna celdas segun ID
    if(node_id==1){
        tsch_schedule_add_link(sf_custom0, LINK_OPTION_RX, LINK_TYPE_NORMAL,
&address_3,4,0,0);
    }else if(node_id==3){
        tsch_schedule_add_link(sf_custom0, LINK_OPTION_RX, LINK_TYPE_NORMAL,
&address_6,3,0,0);
        tsch_schedule_add_link(sf_custom0, LINK_OPTION_TX, LINK_TYPE_NORMAL,
&address_1,4,0,0);
    }else if(node_id==6){
        tsch_schedule_add_link(sf_custom0, LINK_OPTION_RX, LINK_TYPE_NORMAL,
&address_8,2,0,0);
        tsch_schedule_add_link(sf_custom0, LINK_OPTION_TX, LINK_TYPE_NORMAL,
&address_3,3,0,0);

    }else if(node_id==8){
        tsch_schedule_add_link(sf_custom0, LINK_OPTION_RX, LINK_TYPE_NORMAL,
&address_10,1,0,0);
        tsch_schedule_add_link(sf_custom0, LINK_OPTION_TX, LINK_TYPE_NORMAL,
&address_6,2,0,0);
    }else if(node_id==10){
        tsch_schedule_add_link(sf_custom0, LINK_OPTION_TX, LINK_TYPE_NORMAL,
&address_8,1,0,0);
    }else{
    }}
```

Capítulo 6. Análisis de resultados

6.1 Pruebas con Cooja Motes

Para analizar y evaluar los resultados de las simulaciones realizadas según los parámetros señalados en el Capítulo 5, se ha analizado la información detallada en las ventanas Mote Output y Radio Messages (ver 0), con el fin de evaluar y comparar distintos parámetros. Dichos parámetros son: retardo extremo a extremo (End 2 End Delay), PDR, PIAT, DSR, evolución de las colas con respecto al tiempo y por último información relativa al consumo. A continuación, se definen los parámetros a evaluar:

- **Retardo extremo a extremo:** indica el tiempo que tarda un paquete desde que es enviado desde el nodo origen hasta que es recibido por el nodo destino.
- **PDR (Packet Delivery Rate):** indica la razón entre los paquetes recibidos con respecto a los paquetes enviados (en tanto por ciento). Su parámetro complementario es la tasa de pérdidas, que sería 100-PDR (%).
- **PIAT (Packet InterArrival Time):** es el tiempo que transcurre entre la recepción de dos paquetes consecutivos. Es decir, si tenemos el paquete X, con tiempo de llegada t_rx_x, y el paquete X+1, con tiempo de llegada t_rx_x+1, el PIAT correspondiente sería el resultado de aplicar la ecuación (4):

$$PIAT = t_{rx_{x+1}} - t_{rx_x} \quad (4)$$

- **DSR (DeadLine Satisfaction Ratio):** tiene relación con el PIAT, e indica el porcentaje de paquetes cuyo PIAT está por debajo del tiempo esperado de llegada. El tiempo esperado de llegada para cada simulación corresponde al parámetro de packet_period: en las simulaciones realizadas tenemos 70ms, 140ms y 500 ms para cada caso.
- **Tamaño de las colas:** indica el número de paquetes que hay en la cola del nodo a lo largo del tiempo. Para las simulaciones realizadas, el tamaño máximo de la cola es de 64 paquetes.

Como se ha mencionado, para el cálculo de todos estos parámetros únicamente ha hecho falta guardar la información mostrada en las ventanas Mote Output y Radio Messages de cada simulación en archivos de texto para poder computarlos. En este sentido, y para evitar confusiones, se ha seguido la misma nomenclatura para todas las simulaciones a la hora de guardar los ficheros. Por tanto, al guardar el fichero del Mote Output, el archivo se llama:

```
Algoritmo_log_packetperiodms.txt
```

Siendo Algoritmo el utilizado en cada caso (orchestra, amus o minimal), y el packetperiod el que se haya configurado en el caso de simulación observado (70, 140 o 500 ms).

Lo mismo ocurre para la ventana Radio Messages, donde la nomenclatura usada ha sido:

```
Algoritmo_radiolog_packetperiodms
```

Siendo Algoritmo el utilizado en cada caso y el packet period el configurado (70, 140 o 500 ms).

En relación con la metodología y el software utilizado para el cómputo de los datos, cabe señalar que un principio se optó por usar Excel y sus herramientas de texto en columnas, importando los distintos ficheros y estructurando los datos en las columnas para la realización de los cálculos. No obstante, debido al gran volumen de información que se tenía que manejar y analizar, se decidió usar el lenguaje Python y la herramienta Jupyter Notebook.

Jupyter Notebook [23] es un proyecto de código abierto, basado en JSON, surgido en 2014 tras el proyecto IPython Project. Es una aplicación que permite crear y compartir código interactivo, además de manejar grandes cantidades de datos y creación de gráficas y modelados matemáticos de alto nivel de forma interactiva. El lenguaje en el que se rige la aplicación es Python.

Por tanto, para los cálculos se han ido programando distintas funciones en Python cuyo parámetro principal de entrada es la ruta donde se encuentra el fichero objetivo de analizar (aunque como se detallará en posteriores apartados algunas funciones tienen también parámetros de entrada adicionales, además de la propia ruta del archivo).

6.1.1 Retardo extremo a extremo (End2End Delay)

La función relativa a la medición de este parámetro es llamada Retardos. Observando la ventana Radio Messages, se observa que los mensajes que son enviados desde el nodo 10 tienen como longitud 53 Bytes, mientras que los que llegan destino al nodo sink (nodo con ID=1) tienen como longitud 54 Bytes. Estos parámetros son útiles a la hora de hacer el filtrado de los paquetes y obtener solo los que nos son de interés.

Por otro lado, el número de secuencia de paquete es el que hemos introducido como datos dentro del paquete UDP enviado. Sabiendo que los paquetes irán numerados hasta el 500 (es decir, desde 0 a 499), para medir el tiempo en el que se recibe cierto paquete hemos de buscar qué número de secuencia tiene en la recepción.

Todas estas operaciones que se han resumido en esta sección son las que realiza en su totalidad la función Retardos que se ha mencionado. Dicha función escrita en Python se puede consultar en el ANEXO C.

Finalmente, se ejecuta la función explicada un total de 9 veces (el número total de simulaciones realizadas) y se realizan las gráficas de todos los casos analizados.

La Figura 16 muestra los resultados relativos a retardos. Como se observa, vemos como para los tres casos de simulación relativos a AMUS el retardo observado es el mismo (30 ms), además de que no se observan paquetes perdidos. Esto es porque la planificación que se ha realizado previamente asegura que no se produzcan ni interferencias ni colisiones y por tanto las perdidas sean nulas. Además, cabe decir que el retardo es el esperado según la planificación, ya que son los 30 ms los equivalentes a los 3 timeslots que pasan desde que el paquete es transmitido hasta su recepción. Por otro lado, en relación con las simulaciones realizadas con Orchestra vemos diferencias entre los tres casos con diferentes periodos de generación de paquete. Para periodo

de 70 ms, se aprecia un incremento del retardo medio a lo largo de la evolución de la simulación. De hecho, un tiempo tan reducido de generación provoca una mayor carga de tráfico, lo que supone una congestión de las colas, y por tanto implica que los paquetes tarden más en transmitirse. Por el contrario, para los casos siguientes (140 y 500 ms), esto observado anteriormente no sucede y se observa un retardo constante a lo largo de la simulación (350 ms) para ambos casos, que correspondería con 5 slotframes (los paquetes siguen una ruta de 5 saltos). Por último, los casos de Minimal indican una situación distinta a la observada. Como se ha explicado en 3.5.3.1 TSCH-Minimal, en Minimal solo hay una celda compartida para todos los nodos y sin distinción de tráfico. Por tanto, hay alta probabilidad de colisiones e interferencias, y es lo que se observa en las gráficas. Para el caso de 70ms, al igual que observado en Orchestra con 70ms, nos encontramos ante una alta carga de tráfico. Esta carga sumada a lo mencionado con respecto a las colisiones de Minimal, provoca que se pierdan gran cantidad de paquetes (vemos el símbolo de paquete perdido -cruz roja- bastantes veces repetidas), además del retardo que va aumentando a lo largo de la simulación. Un menor número de pérdidas se observa para el caso de 140 ms, ya que se alivia la carga de tráfico, pero sigue persistiendo la naturaleza de colisiones de Minimal. Finalmente, se aprecia en la situación de los 500 ms las casi nulas pérdidas de paquetes y un retardo medio constante (210 ms), que correspondería con la duración de 3 slotframes (ruta de 4 saltos).

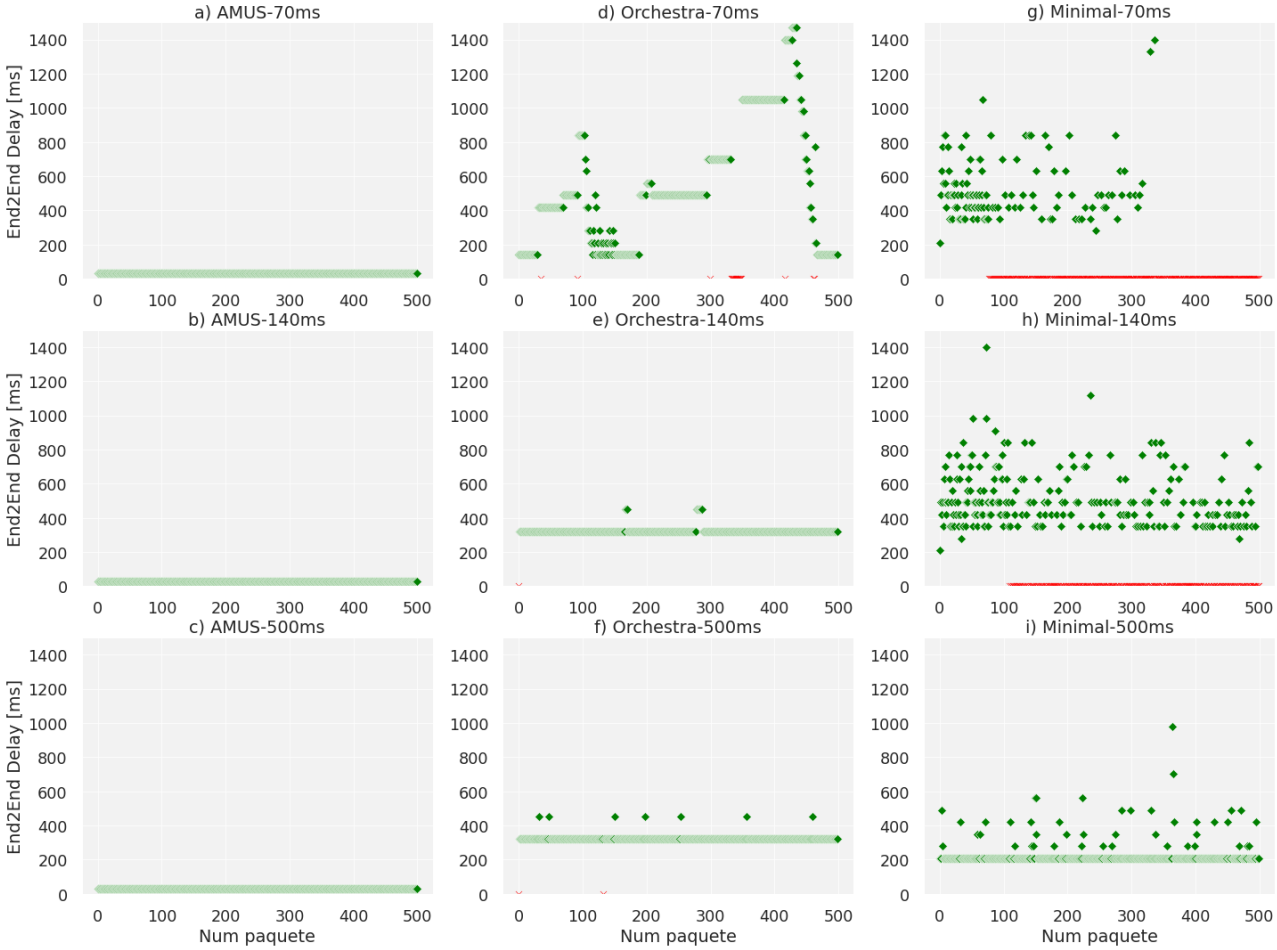

Figura 16: Retardo extremo a extremo de las planificaciones

6.1.2 PDR (Packet Delivery Rate)

Se ha explicado que es el porcentaje de paquetes recibidos respecto de los enviados. La ecuación (5) indica cómo se calcula ese valor de PDR:

$$PDR\ (\%) = \left(\frac{num_{paquetes_{recibidos}}}{num_{paquetes_{enviados}}}\right) \cdot 100 \quad (5)$$

Donde $total_{enviados}$ varía para cada caso de simulación, y por tanto se debe parametrizar en la función en Python. Dicha función se denomina PDR, y se puede consultar en su totalidad en el ANEXO C.

La Figura 17 muestra el valor de PDR para cada método de planificación simulado y los diferentes tiempos de generación de paquete. Se puede observar la correlación entre esta gráfica y la Figura 16, relativa al retardo extremo a extremo. Se aprecia como tanto para AMUS como para Orchestra el valor del PDR es prácticamente 100 % (en AMUS se consigue la totalidad de recepción, mientras que en Orchestra se observa apenas el 0.2 % de pérdidas para la situación de 500 ms). Por último, solo cabe comentar los valores de Minimal, que muestran un bajo nivel de recepción para el análisis de los paquetes cada 70 y 140 ms (de hecho, solo se observa un 30 % y un 50 % aproximadamente de recepción). Por el contrario, para el último caso sí que se consigue la nulidad de pérdidas. Esto se ve justificado por las mismas razones expuestas en la sección 5.1.2 y 6.1.1, que indican que la congestión de los distintos nodos junto a que Minimal es propenso a las colisiones, producen una alta tasa de pérdidas en la red.

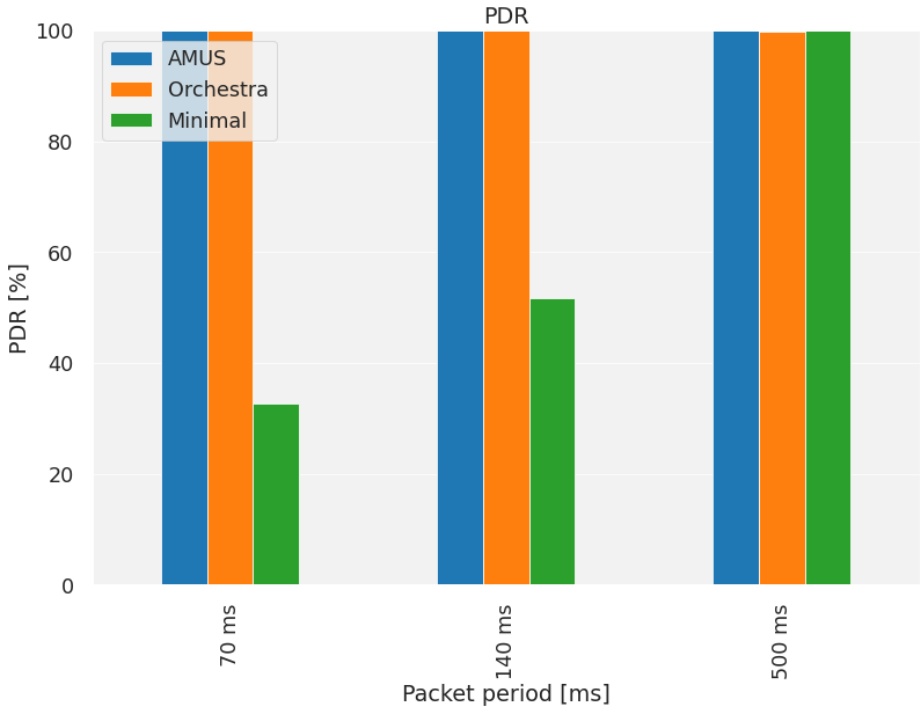

Figura 17: PDR en los diferentes planificadores TSCH

6.1.3 PIAT (Packet InterArrival Time)

Este parámetro indica también cómo de congestionada y cargada está la red durante la transmisión. Como se ha explicado en la introducción del Capítulo 6, el PIAT indica el tiempo entre paquetes recibidos consecutivos. En este sentido, la función de Python responsable del cálculo de este parámetro se denomina InterArrival, y, al igual que todas las demás, se puede consultar en su totalidad en el ANEXO C.

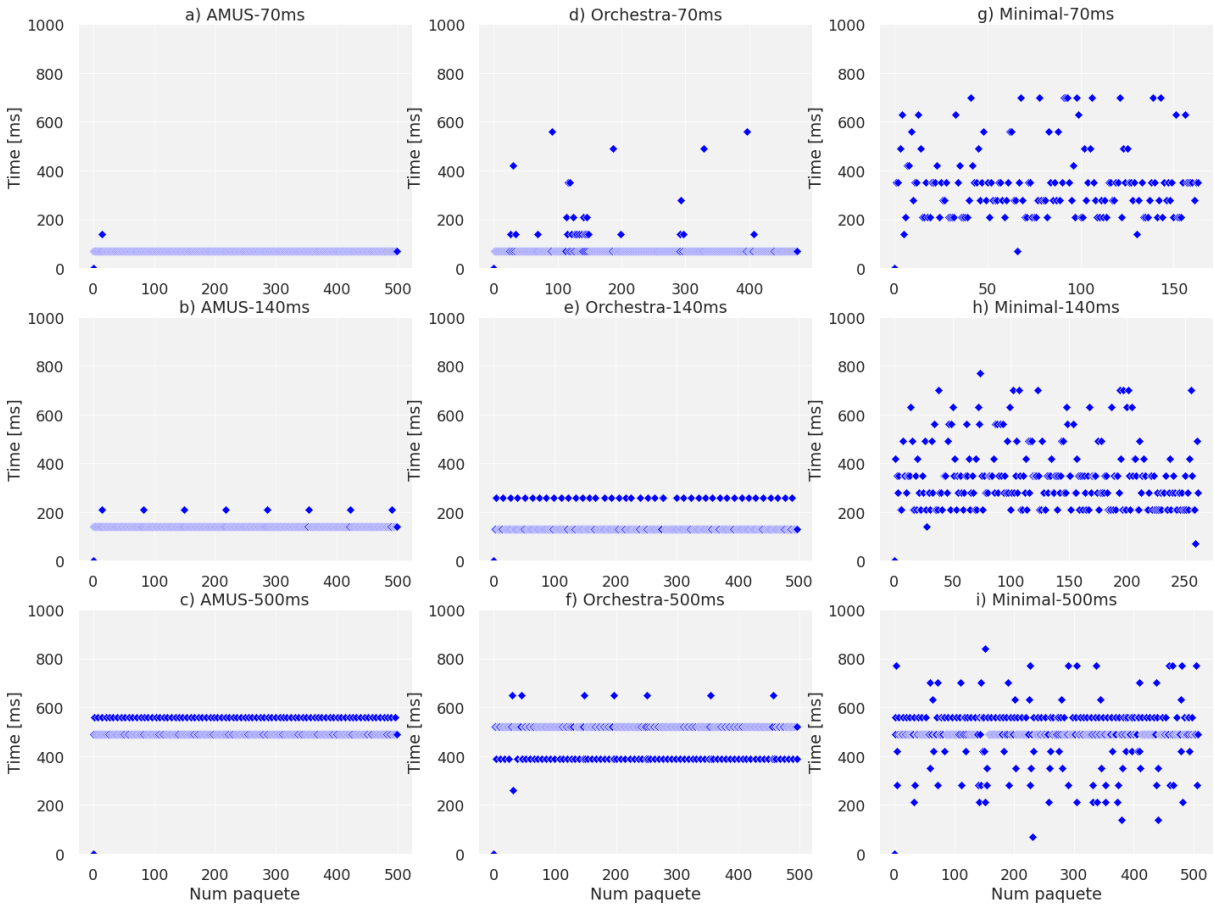

Figura 18: Representación gráfica del PIAT

La Figura 18 muestra los valores del PIAT para los casos analizados. Si se observan todas las situaciones con el periodo de paquetes cada 70 ms se aprecia que para AMUS y Orchestra los valores son los esperados (70 ms), ya que al no producirse sobrecarga y congestión de las colas los paquetes llegan a la misma frecuencia que son enviados (1 paquete cada 70 ms). Lo mismo se puede aplicar para los otros dos periodos (140 y 500 ms), lo que demuestran que ambos métodos de planificación no producen sobrecarga en el desarrollo de la transmisión. Por otro lado, para Minimal se aprecia una mayor variabilidad de los valores de este parámetro temporal, y en comparación con los otros protocolos se observa que estos valores son superiores. Las múltiples colisiones provocan que se realicen sucesivos reintentos (retransmisiones) y por tanto los paquetes llegan al destino más espaciados en el tiempo, de ahí los valores incrementados con respecto a los otros.

6.1.4 DSR (Deadline Satisfaction Ratio)

Utilizando los resultados obtenidos en 6.1.3, se realiza un filtrado de todas las series de datos obtenidas con datos de PIAT solo guardando los valores por debajo del valor de periodo de paquete (70, 140 y 500 ms para las situaciones correspondientes). El cómputo de los datos da como resultado la gráfica de la Figura 19. Se aprecian unos altos valores para los casos de AMUS y Orchestra para los periodos de generación de paquetes de 70 y 140 ms (ligeramente superior AMUS a Orchestra). En la situación de los 500 ms, los valores de DSR son inferiores a todos los demás, destacando el relativo a Orchestra (alrededor de 15 %), aunque se destaca en esa última tanda el valor relativo a Minimal (80 % de DSR), ya que en las dos situaciones anteriores (70 y 140 ms) el valor de DSR es casi nulo.

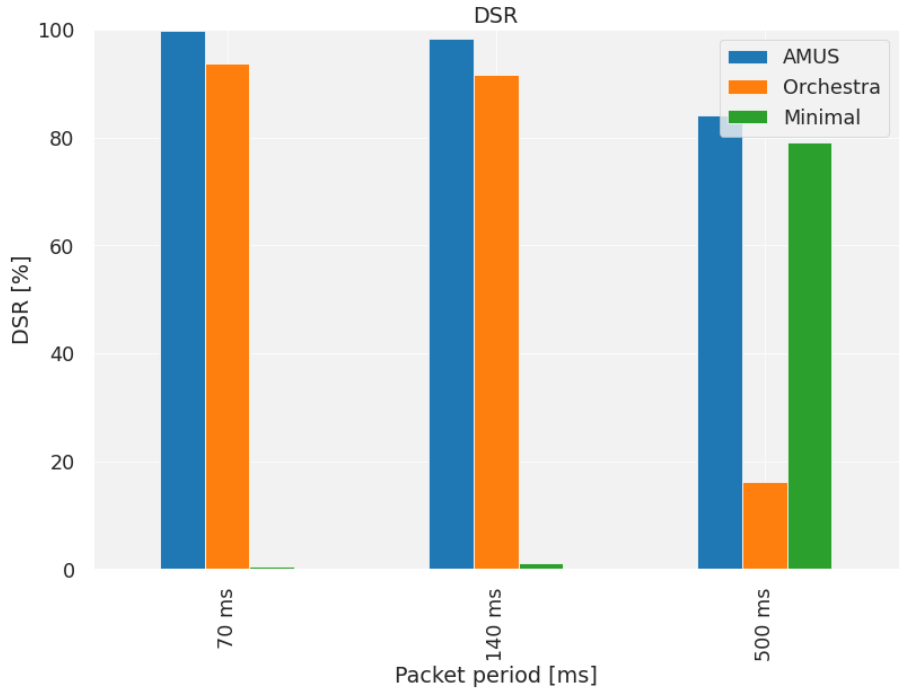

Figura 19: Valores de DSR para los casos de simulación

6.1.5 Evolución de las colas

Para el estudio de cómo evoluciona el tamaño de las colas en el nodo 10 (nodo origen del tráfico), en el código de este se creó un hilo externo en el que se muestra por salida el valor instantáneo del número de paquetes en cola. Esto se ha conseguido mediante el uso de la función que se proporciona en la API de Contiki. Dicha función es tsch_queue_global_packet_count(), y se ha programado para que se imprima el mensaje cada 70 ms (duración del slotframe).

Mediante la función `colas`, cuyo contenido se puede consultar en el ANEXO C, realiza el cómputo y la evolución del tamaño de las colas y guarda los datos en una variable. Posteriormente, en un script se realizan las gráficas que representan esa evolución a lo largo del tiempo, y cuyos resultados se muestran en la Figura 20.

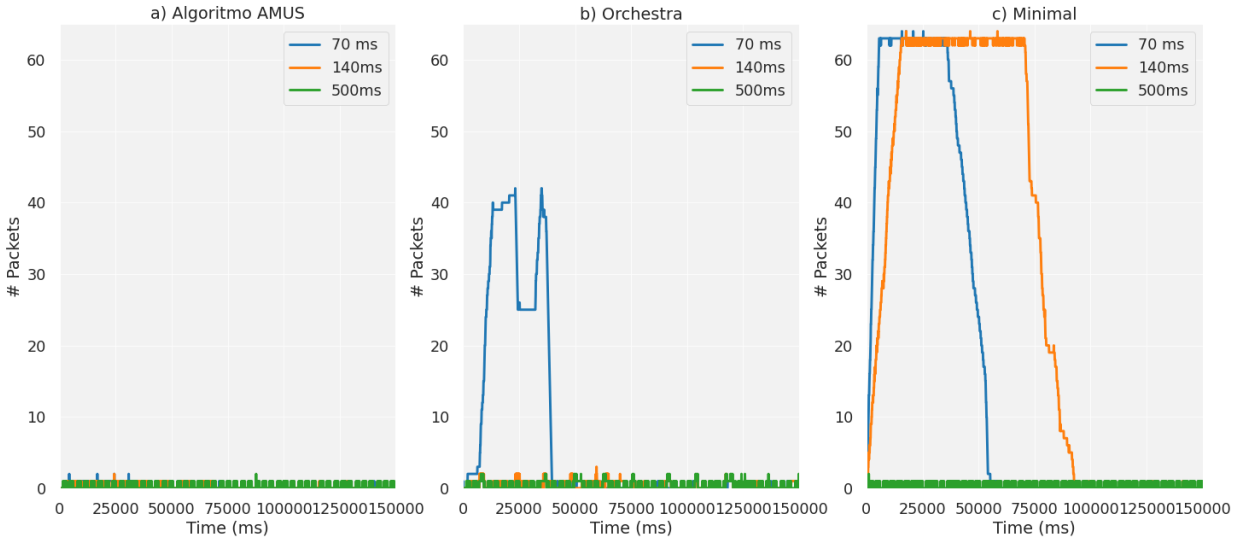

Figura 20: Evolución de las colas en el tiempo de simulación

Como podemos apreciar en la Figura 20, los tres algoritmos de planificación presentan comportamientos distintos en cuanto a la congestión se refiere. Por un lado, para las transmisiones con el algoritmo que presentan en AMUS [19], apenas hay paquetes en cola durante la simulación (se observan alguno, pero no tiene repercusión).

Por otro lado, si se observan las evoluciones de Orchestra, la curva que sorprende es la relativa al periodo de envío de 70 ms. De hecho, en los primeros instantes se incrementa el número de paquetes rápidamente, debiéndose a la alta carga de tráfico que supone tan alta frecuencia de envío de paquetes, ya que en Orchestra un nodo solo recibe un paquete de datos cada slotframe. Más adelante, el tamaño desciende hasta los 25 paquetes y se mantiene para finalmente volver a los 40 (máximo anterior) y descender por último hasta vaciarse del todo. Esta gráfica traduce también lo que se observó en la Figura 16d), donde vemos el retardo para Orchestra con envíos cada 70ms que se mantiene constante al inicio, se incrementa para posteriormente reducirse y finalmente realizar otro pico y mantenerse constante. Para las otras dos situaciones de Orchestra tan solo cabe decir la nula evolución de las colas, manteniéndose prácticamente vacías.

Por último, destacamos en Minimal las curvas relativas a los 70 y los 140ms. Como ya se ha mencionado antes, la alta frecuencia de envío de 70ms provoca una mayor carga de tráfico. Todo eso, sumado a las características de Minimal con su alta probabilidad de colisión debido a la única celda planificada disponible, provoca el rápido llenado de la cola (máximo de 64 paquetes). La congestión de las colas, junto a los intentos de retransmisión de TSCH de los paquetes se traduce en el bajo valor de PDR observado en la Figura 17 y los elevados retardos observados en la Figura 16. Para la transmisión cada 140 ms no se observa una situación tan extrema como la anterior descrita ya que no hay tanta carga de tráfico, aunque sí que se llegan a llenar las colas, pero no se producen tantas pérdidas, de ahí que el valor de PDR sea superior al anterior. Finalmente, con los 500 ms no se encuentran problemas ya que la red no se encuentra con la misma intensidad de tráfico de las otras situaciones y por tanto no hay alteraciones en los buffers y se consigue un 100 % de fiabilidad.

Habiendo analizado de forma global las prestaciones de los métodos presentados, aplicados en un escenario "ideal" (se utiliza el calificativo "ideal" ya que los Cooja Motes usados para esas pruebas son dispositivos virtuales y no emulan ningún dispositivo real), para observar uno de los principales factores en redes IWSN, la temporización.

En la próxima sección se describirán los resultados obtenidos con relación a las pruebas realizadas con los Z1, de tal forma que se puede observar un resultado realista y además de analizar el factor de la temporización, se estudiará el rendimiento relativo al consumo energético de dichos nodos (usando la herramienta Simple-Energest 4.2.1). Por tanto, se utiliza el banco de pruebas en su máximo potencial.

6.2 Pruebas con Zolertia Z1 Motes

Tras haber descrito las principales características de los nodos de este fabricante, se ha realizado el consiguiente análisis tras las pruebas llevadas a cabo.

Cabe señalar que la capacidad de memoria de estos dispositivos es limitada, y por tanto todo lo relacionado con el tamaño de los buffers y almacenamiento es menor con respecto a los Cooja Motes, ya que estos no tienen ninguna limitación con respecto al almacenamiento. Por esa razón, en esta parte de las pruebas solo se ha analizado con Orchestra y Minimal. Recordemos que cuando nos referimos a AMUS no se está realizando la planificación tal cual viene explicado en [19], se calcula la planificación según el algoritmo expuesto y se introduce manualmente usando la API disponible en Contiki. Esa planificación introducida manualmente produce una sobrecarga y un peor rendimiento durante la transmisión (en la simulación). Por estas razones se han realizado solo las pruebas con Orchestra y Minimal y se compararán los resultados con los obtenidos anteriormente (Cooja Motes).

En este caso los parámetros a analizar, siguiendo los mismos procesos que se han explicado en la sección anterior, son:

- **Retardo extremo a extremo (End2End Delay)**
- **PDR**
- **PIAT**
- **Consumo energético:** este punto es clave a la hora de la comparación, además de la temporización. Y es que actualmente, como ya se ha mencionado, lo que se busca en redes WSN es robustez y un consumo eficiente. Mediante estos resultados (usando la extensión Simple-Energest) se puede comparar ampliamente el rendimiento de los dos algoritmos que se pretende estudiar.

6.2.1 Retardo Extremo a Extremo (End2End Delay)

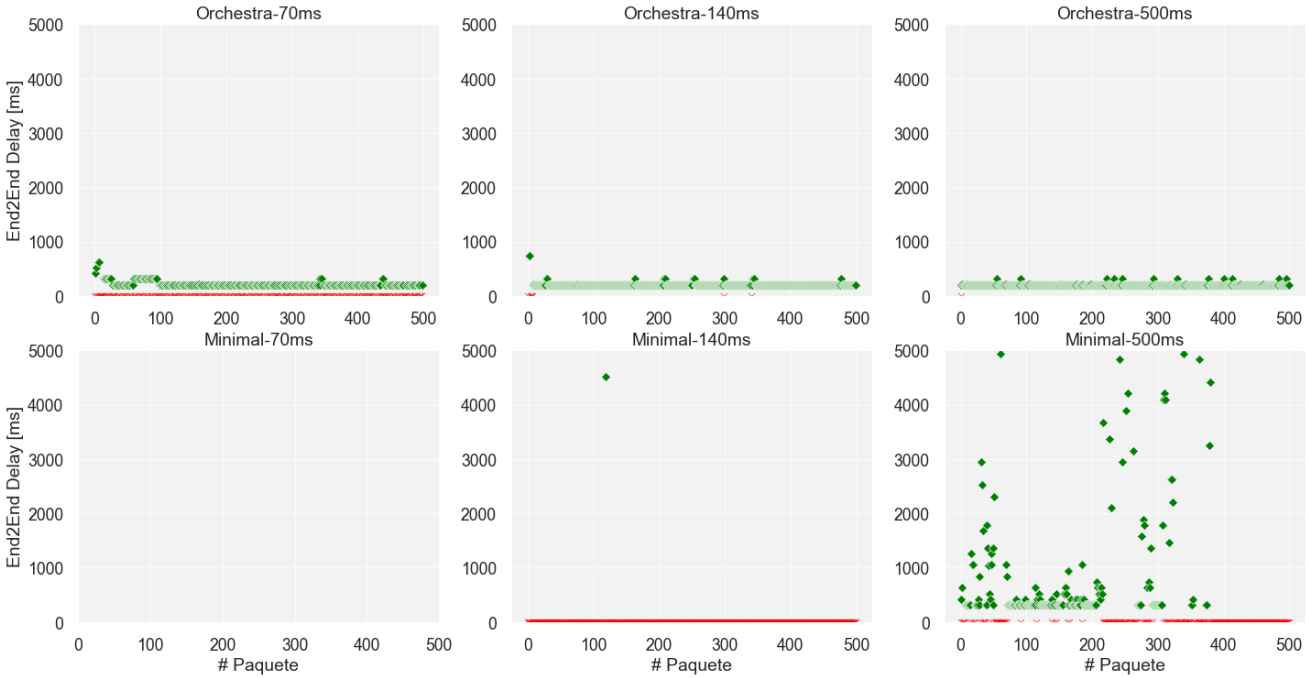

Figura 21: Retardo extremo a extremo en los nodos Z1

La Figura 21 muestra el retardo extremo a extremo para las simulaciones. Destacamos un similar comportamiento al análisis con Cooja Motes (Figura 16) y vemos como los retardos en Orchestra son prácticamente constantes. Solamente destacar algunas pérdidas (para el caso de los 70 ms) que son debidas a la alta carga de tráfico de la red.

En Minimal la situación es complatemente distinta. Para mensajes cada 70 ms y 140 ms la situación es inabordable, por las características ya explicadas de Minimal y además la restricción de memoria que supone esta emulación real de los nodos. Solamente se observa en los 500 ms una transmisión satisfactoria, aunque aun se observan múltiples paquetes pérdidos.

6.2.2 PDR

La Figura 22 muestra el PDR de los casos analizados. Se observa claramente la alta fiabilidad y operabilidad de Orchestra frente a Minimal. Destacar el valor de PDR nulo para el caso de Minimal en 70 ms.

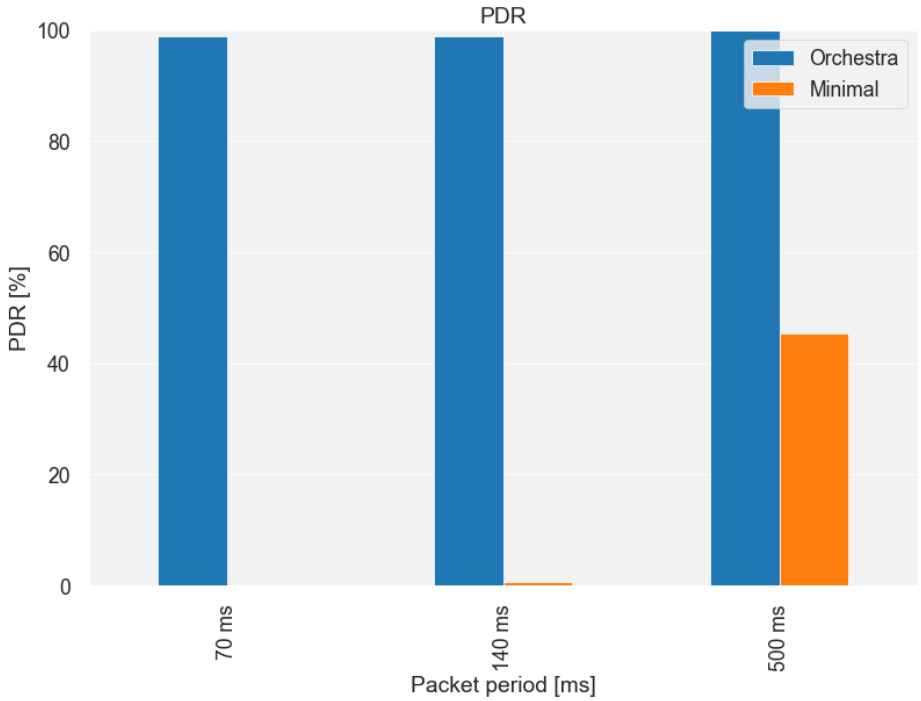

Figura 22: PDR en nodos Z1

6.2.3 PIAT

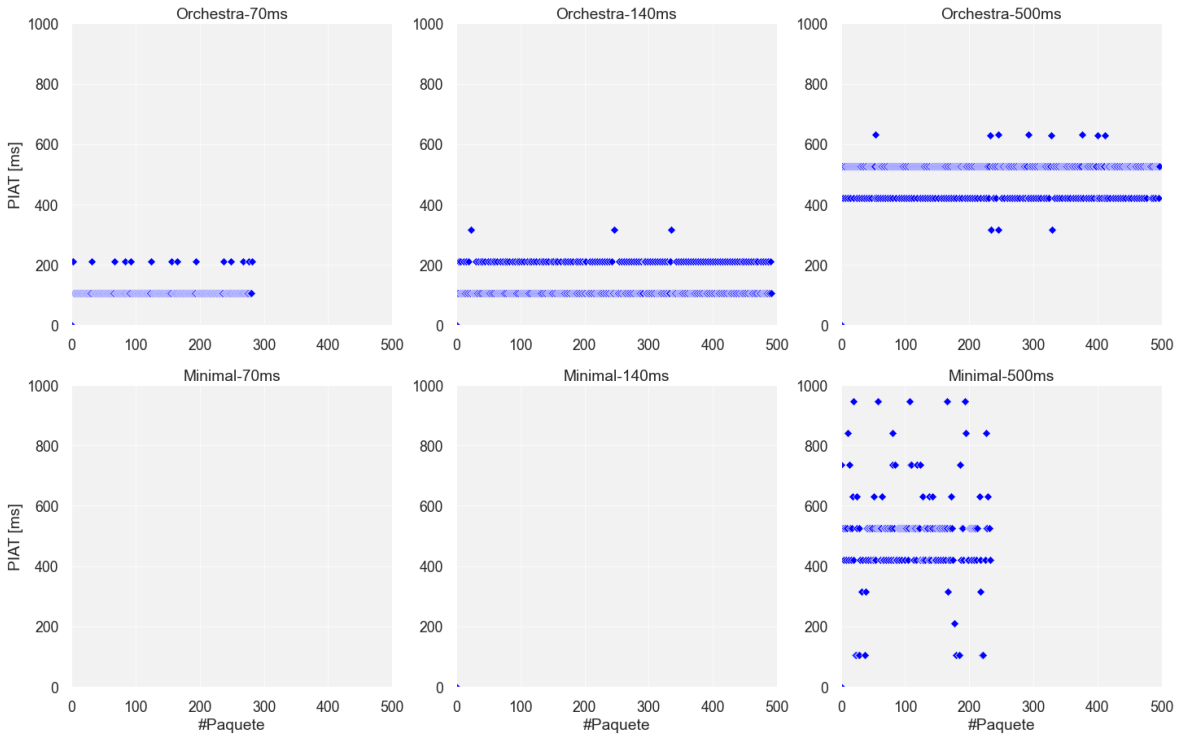

Figura 23: PIAT para los nodos Z1

6.2.4 Consumo energético

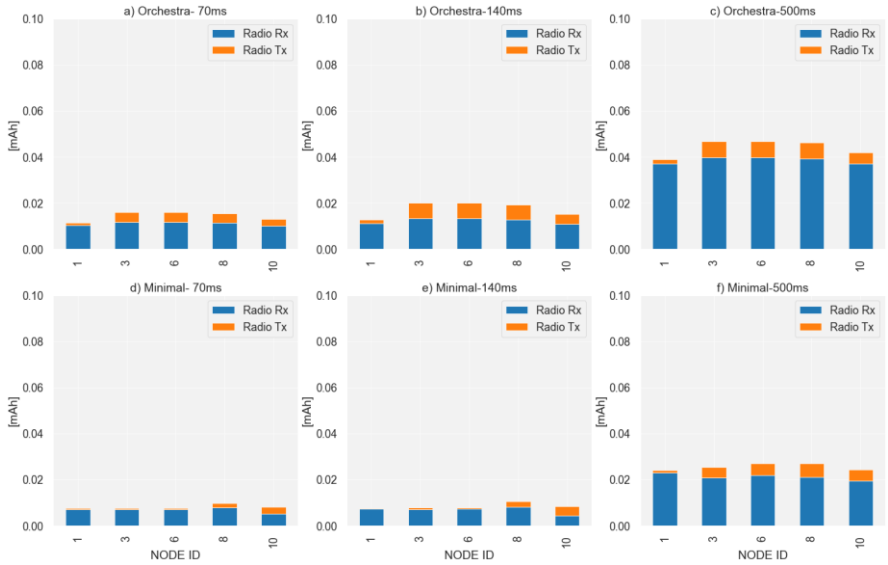

Figura 24: Consumo energético en las simulaciones

Por último, se ha analizado, gracias a la herramienta Simple-Energest explicada en 4.2.1, el consumo de los nodos. Con los resultados obtenidos, y con los datos proporcionados por el fabricante (ver Tabla 4), se ha calculado el consumo correspondiente. En la Figura 24 se muestra el consumo (en mAh) relativo a los nodos que conforman la ruta que han seguido los paquetes transmitidos.

En cuanto a los valores, si comparamos ambos protocolos a modo general se observa un menor consumo por parte de Minimal con respecto a Orchestra. La planificación TSCH es la razón principal de esta diferencia. Mientras en Orchestra se activa la radio varias veces en un slotframe, en Minimal se activa únicamente una. Esto tiene su ventaja a nivel de eficiencia energética (para Minimal, obviamente), mientras que el principal inconveniente es la eficiencia de las transmisiones (como ya se ha visto en la Figura 21 y la Figura 22).

Capítulo 7. Conclusiones y propuesta de trabajo futuro

Tras haber analizado todas las prestaciones de los distintos algoritmos de planificación TSCH, se han podido apreciar las distintas diferencias entre cada uno.

Por un lado, Minimal ofrece más ventajas en cuanto a temporización y consumo de recursos se refiere, debido a su sencillez y fácil configuración. Sin embargo, a nivel de fiabilidad no destaca, encontrándose así los altos valores de tasa de pérdidas. Orchestra en cambio sí que ofrece un alto nivel de fiabilidad (hemos observado el alto nivel de tasa PDR en la Figura 17) aunque presenta un ligero incremento en el retardo (cabe señalar que un nodo solo recibe un mensaje cada slotframe -en nuestro caso son 70 ms). Por último, la simulación realizada con la planificación según el algoritmo descrito en AMUS [19] demuestra la alta precisión que se debe tener con respecto a la planificación de las rutas, lo que supone un excelente nivel de fiabilidad (las pérdidas obtenidas son nulas) y un retardo mínimo (los 30 ms que se observan en la Figura 16 son los esperados según la ruta de 3 saltos). Por tanto, los beneficios de los algoritmos (Orchestra y Minimal) se pueden aprovechar dependiendo el contexto de la aplicación. Si tenemos unos altos requisitos de fiabilidad y temporización, es evidente que Orchestra es uno de los métodos apropiados (de hecho, es de los más usados actualmente); mientras que si lo que se busca es una eficiencia en el consumo de energía y temporización Minimal es más adecuado. Las diferencias entre ambos algoritmos se detallan en la Tabla 6.

Por todas estas razones, es evidente que a la hora del diseño y la gestión de redes WSN, la planificación de la transmisión es uno de los pilares fundamentales, ya que la gran cantidad de datos que son manejados, como las características de los entornos de las principales aplicaciones, en especial los entornos industriales (hablamos de las IWSN), en los que los requerimientos de temporización y fiabilidad son muy elevados. Hablamos pues del concepto de redes deterministas, la garantía de un buen rendimiento gracias al conocimiento previo de las características de la red. Además, todo lo expuesto coincide con el auge de los últimos tiempos de la digitalización de la sociedad y la aparición de la Industria 4.0.

Finalmente, como propuesta de trabajo futuro, destacamos la aplicación del paradigma de las redes definidas por software (SDN) en el mundo de las redes WSN. Las SDN se caracterizan por la figura del controlador, quien realiza las tareas de gestión y control de la red y por tanto los nodos solo deben realizar tareas de encaminamiento de paquetes y consultas previas al controlador sobre su funcionamiento. En [5] y [24] describen el funcionamiento y las ventajas que supone la aplicación de este concepto en las redes inalámbricas, además del protocolo SD-WISE, diseñado para su funcionamiento, por lo que sería interesante el estudio y el análisis con esta nueva ideología de red para comparar con los resultados obtenidos en este proyecto y los beneficios que suponen en el campo de los procesos industriales, como se destaca en las investigaciones en [25].

Tabla 6: Comparativa de los algoritmos tras los resultados obtenidos

	TSCH-MINIMAL	**ORCHESTRA**	
Formación/Construcción de la topología	Protocolo RPL	Protocolo RPL	
Asignación y distribución de celdas	Una única celda compartida para todos los nodos	**Receiver-based** -EB Slotframe -CS Slotframe -RBS Slotframe	**Sender-based** - EB Slotframe -CS Slotframe -SBS Slotframe
Latencia (Retardos)	Bajo retardo	Mayor retardo	
Fiabilidad (PRR)	Baja PDR (pérdidas por colisiones o congestión de colas)	Alta fiabilidad. Planificación sin colisiones	
Consumo	Menor consumo. Todos los nodos se activan, una vez por slotframe	Mayor consumo. Los nodos se activan varias veces, aunque no tengan paquetes para transmitir	

Capítulo 8. Referencias

[1] [En línea]. Available: https://www.aner.com/blog/industria-4-0.html.

[2] M. Gracia, «IoT- Internet Of Things,» Deloitte España, [En línea]. Available: www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html. [Último acceso: Mayo 2021].

[3] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Internet_de_las_cosas.

[4] [En línea]. Available: https://www.powerdata.es/big-data.

[5] H. Mostafaei y M. Menth, «Software-defined wireless sensor networks: A survey,» *Journal of Network and Computer Applications 119,* pp. 42-56, 2018.

[6] A. Koubaa, M. Alves y E. Tovar, «IEEE 802.15.4 for Wireless Sensor Networks. A Technical Overview,» IPP Hurray.

[7] «Aprendiendo Arduino: Aprendiendo a manejar Arduino en profundidad,» [En línea]. Available: https://aprendiendoarduino.wordpress.com/tag/ieee-802-15-4/.

[8] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Protocolo_de_enrutamiento_RPL.

[9] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur y R. Alexander, «RPL: IPv6 routing protocol for low-power and lossy networks,» RFC EDITOR, 2012.

[10] D. De Guglielmo, S. Brienza y G. Anastasi, «IEEE 802.15.4e: A Survey,» *Computer Communications* , vol. 88, pp. 1-24, 2016.

[11] A. Karalis, D. Zorbas y C. Douligeris, «Collision-Free Advertisement Scheduling for IEEE 802.15.4-TSCH Networks,» *Sensors*, 2019.

[12] S. Kim, H.-S. Kim y C. Kim, «ALICE: Autonomous Link-based Cell Scheduling for TSCH,» de *ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2019.

[13] A. Elsts, S. Kim, H.-S. Kim y K. Chongwon, «An Empirical Survey of Autonomous Scheduling Methods for TSCH,» *IEEE Access*, pp. 67147-67165, 21 April 2020.

[14] S. Duquennoy, A. Elsts, B. Al Nahas y G. Oikonomou, *TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation.*

[15] B. Al Nahas, O. Landsiedel, S. Duquennoy y T. Watteyne, «Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH,» 2015.

[16] T. van der Lee, G. Exarchakos y A. Liotta, «Distributed TSCH Scheduling: A Comparative Analysis,» de *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Banff, Canada, 2017.

[17] R. Hwang, C. Wang y W. Wang, «A Distributed Scheduling for IEEE 802.15.4e Wireless Sensor Networks,» *Computer Standards & Interfaces,* vol. 52, pp. 63-70, 2017.

[18] S. Rekik, N. Baccour, M. Jmaiel y K. Drira, «A Performance Analysis of Orchestra Scheduling for TSCH Networks,» *Internet Technology Letters,* vol. 1, nº 3, 2017.

[19] Y. Jin, P. Kulkarni, J. Wilcox y M. Sooriyabandara, «A Centralized Scheduling Algorithm for IEEE 802.15.4e TSCH based Industrial Low Power Wireless Networks,» de *IEEE Wireless Communications and Networking Conference* , 2016.

[20] G. Oikonomou, «Contiki-ng,» 8 Diciembre 2020. [En línea]. Available: github.com/contiki-ng/wiki.

[21] A. Elsts. [En línea]. Available: https://github.com/contiki-ng/contiki-ng/wiki/Instrumenting-Contiki-NG-applications-with-energy-usage-estimation.

[22] Z. Z. Datasheet. [En línea]. Available: http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf.

[23] «Jupyter,» [En línea]. Available: https://jupyter.org/about.

[24] A.-C. Anadiotis, L. Galluccio, S. Milardo, G. Morabito y S. Palazzo, «SD-WISE: A Software-Defined WIreless SEnsor network,» *Computer Networks,* pp. 84-95, 2019.

[25] F. Orozco Santos, V. Sempere Payá, T. Albero Albero y J. Silvestre Blanes, «Enhancing SDN WISE with Slicing Over TSCH,» *Sensors,* vol. 21, nº 1075, pp. 1-28, 2021.

[26] M. Mohamadi, B. Djamaa y M. R. Senouci, «Performance Evaluation of TSCH-Minimal and Orchestra Scheduling in IEEE 802.15.4e Networks,» *IEEE Journal.*